Statistical Applications in Geographical Health Studies

José Miguel Martínez Martínez
PhD Student

Joan Benach de Rovira
Universitat Pompeu Fabra
PhD Advisor

Yutaka Yasui
University of Alberta
PhD Advisor

Josep Ginebra i Molins
Universitat Politècnica de Catalunya
PhD Tutor

Doctorate in Technical and Computer Applications of Statistics,
Operational Research and Optimization.
Universitat Politècnica de Catalunya
Barcelona, 8/05/2006
CHAPTER 7

Individual and aggregated health outcomes studies

"If exposure to a necessary agent is homogeneous within a population, then case/control and cohort methods will fail to detect it"

Geoffrey Rose

7.1 Introduction.

Studies based on individual data permit the assessment of the relationship between a health outcome (disease or death) and a series of characteristics (risk factors or confounding factors) measured at individual level. In other words for each individual we have information on their health outcome and their risk factors and confounding factors. However, as mentioned in chapter 1, when the individuals are hierarchically organized in groups, for example neighborhoods of a city, the studies using individual data must consider this hierarchical structure in order to: 1) take account of the relationship of dependence of the individuals within each group to avoid obtaining incorrectly significant relation between exposure factors and disease when these do not exist and 2) incorporate the influence of contextual or group factors into the study of health.124

In order to deal, for the purposed of health studies, with the organization of individuals into groups, statistical models incorporating random effects may be used. Through random effects models it is possible to simultaneously combine the study of the influence of both individual and group factors on individual health outcomes,
controlling for the dependency of the individuals within each group. Random effects models, also known as multilevel models in other contexts, are also appropriate when our objective is not simply to assess the relationship between exposure factors and health outcome, but also when it is proposed to study and explain variations in health outcome within and between groups. Furthermore, as mentioned in the first part of this thesis, in the study of small areas, random effects models are used to control for the variability in estimated health indicators.

However, analyses of individual disease-exposure data within a population are useful when exposure of interest varies sufficiently within the population. When the within-population variance of exposure is limited, however, power of the individual-data analysis within a population is reduced. As Geoffrey Rose pointed out “If exposure to a necessary agent is homogeneous within a population, then case/control and cohort methods will fail to detect it.” Dietary and enviromental factors provide examples that can involved limited ranges within populations available for study but with a significant variability between population groups. In such situations, aggregated health data studies over different populations can be used. Specifically, aggregated-data analyses of disease data across populations proposed by Prentice and Sheppard, with a sample of individual exposure data from populations, can be powerful in estimating the exposure effect if between-population variation of exposure is large. Individual and aggregated-data analyses approaches are useful depending on where the exposure variation exists.

In the following section we describe the individual random effects model (IRM) and the aggregated random effects model (ARM) proposed by Prentice and Sheppard for obtaining relative rates of disease. The ARM will also be compared with the classical ecological random effects model (ERM). Finally, we will explain the process for estimating the IRM and ARM using the estimating equation approach.

7.2 Relative rate analysis of individual- and aggregated-data.

This section reviews the individual- and aggregated-data models, following Prentice and Sheppard’s work.
7.2.1 The individual-data model.

Let \( p_{ki} \) denote the probability that the \( i \)th individual in the \( k \)-th population, with size \( n_k \) (\( k=1,\ldots,K \)), develops a certain disease within a defined follow-up period. We consider a relative rate model:

\[
p_{ki} = p_{k0} e^{z_{ki}^T \beta}
\]

where \( z_{ki} \) is a vector of covariates, \( p_{k0} \) is a ‘baseline’ disease probability for the \( k \)-th population corresponding to \( z_{ki} = 0 \) and \( \beta \) is a parameter vector to be estimated. The random effects assumption gives:

\[
p_{k0} = h_k e^{\gamma_0}
\]

where \( e^{\gamma_0} \) denotes the expected baseline rate and \( h_k \) denotes the residual baseline rate or ‘frailty’ of the \( k \)-th population. We consider that \( h_k \)'s are independent random effects with mean 1 and variance \( \sigma^2 \). Under the random effects assumption, the model can be written as

\[
p_{ki} = h_k e^{x_{ki}^T \alpha}, \tag{7.1}
\]

where \( x_{ki} = (1, z_{ki}^T) \) and \( \alpha^T = (\gamma_0, \beta^T) \).

The model expressed in (7.1) can be estimated considering that we know the covariate information in all the individuals of the \( k \)-th cohort, i.e. in the \( n_k \) individuals. Such covariate information usually is not available and we can consider the model in terms of a covariate sample size \( m_k \) in the \( k \)-th cohort.

7.2.2 The aggregated-data model.

An aggregate-data model as defined by Prentice and Sheppard, can be induced from the random effects model for individual data by averaging \( h_k e^{x_{ki}^T \alpha} \) over the \( n_k \)
individuals within \( k \)-th population, and considering the average disease probability \( \bar{p}_k \) of the population among the \( n_k \) individuals:

\[
\bar{p}_k = h_k \left( \sum_{i=1}^{n_k} e^{\alpha_i} / n_k \right)
\]

Following Prentice and Sheppard’s notation we can express the aggregated data model as

\[
\bar{p}_k = h_k \varepsilon_{n_k} \{ e^{\alpha_i} \} \tag{7.2}
\]

where \( \varepsilon_{n_k} \{ a_k \} = n_k^{-1}(a_{k1} + \ldots + a_{kn_k}) \) denotes the average of the argument over the \( n_k \) individuals in the \( k \)-th population.

The model expressions presented consider that we know the covariate information in all the individuals of the \( k \)-th cohort, i.e. in the \( n_k \) individuals. As we pointed out in the section 7.2.1, such covariate information is not usually available and we can express the models in terms of a covariate sample size \( m_k \) in the \( k \)-th cohort. In the same way, an aggregate-data model can be induced from the random effects model for individual data by averaging \( h_k e^{\alpha_i} \) over the \( m_k \) individuals in the sample within \( k \)-th population, and considering the average disease probability \( \bar{p}_k \) of the population among the \( n_k \) individuals:

\[
\bar{p}_k = h_k \varepsilon_{m_k} \{ e^{\alpha_i} \} \tag{7.3}
\]

where \( \varepsilon_{m_k} \{ a_k \} = m_k^{-1}(a_{k1} + \ldots + a_{km_k}) \) denotes the average of the argument over the \( m_k \) individuals in the \( k \)-th population. Note that the left-hand side of the equation is the average based on the aggregated data (i.e., \( n_k \) individuals), while the right-hand side is the average based on the individual data (i.e., \( m_k \) individuals). Under the assumption that the individual samples are random samples of a sufficient size from the population, the aggregated-data model holds.
7.3 Differences between aggregated data and ecological studies.

Aggregated data and ecological studies are different, even though both are group-level studies that use aggregate health outcomes\textsuperscript{31,132}. In this section, we show the difference between the ARM in (7.2) and the ecological random effects model (ERM), commonly used in small areas studies.

An ecological random effects model can be induced from the individual data by averaging $x_{ki}$ over the $n_k$ individuals within k-th population, instead of averaging $e^{x_{i}^{T}x_{i}^{T}x_{i}}$ as in the ARM, and considering the average disease probability $\overline{p}_k$ of the population among the $n_k$ individuals again just as for the ARM:

$$\overline{p}_k = h_k e^{x_{i}^{T}x_{i}}$$

where $x^{T} = \frac{\sum x_{ki}}{n_k}$. As has been shown, the ARM proposed by Prentice and Sheppard arises from the aggregation of that component of the IRM which contains the covariates and represents the relationship of these covariates with the health outcome, whereas ecological studies only consider averages of individual covariates or group variables. For more details on ecological studies the reader is referred to chapter 1 and 4.

7.4 Relative rate inference for individual and aggregated random effects models based on estimating equations.

We consider individual data for a random sample of $m_k$ individuals on K populations with population sizes $n_k$'s ($k=1,...,K$). We denote the disease outcome variable of the i-th individual in the k-th population as $Y_{ki}$. The variable $Y_{ki}$ takes a value of one if the outcome of interest (disease/death) occurs on the individual within the defined study follow-up period, and zero otherwise. We denote $Y_{k}^{T} = (Y_{k1},...,Y_{km_k})^{T}$ and $\mu_{k}^{T} = (\mu_{k1},...,\mu_{km_k})^{T}$ with
\[
\mu_{ki} = E(Y_{ki}) = E(E(Y_{ki} | h_k)) = e^{x_{ki}^T \alpha}.
\]

Then the individual-data estimating equation for model (7.1) is \textsuperscript{128,133}

\[
\sum_{k=1}^{K} (D_k^i)^T (V_k^i)^{-1} (Y_k^i - \mu_k^i) = 0, \quad (7.4)
\]

where \(D_k^i = \partial \mu_k^i / \partial \alpha^T\), \(V_k^i = \Delta_k + \sigma^2 \mu_k^i (\mu_k^i)^T\) with \(\Delta_k = \text{diag}[\mu_{ki} \{1 - (1 + \sigma^2) \mu_{ki}\}]\).

The inverse of the variance-covariance matrix \((V_k^i)^{-1}\) can be computed \textsuperscript{128,134} by

\[
(V_k^i)^{-1} = (\Delta_k)^{-1} - \sigma^2 (\Delta_k)^{-1} \mu_k^i (\mu_k^i)^T (\Delta_k)^{-1} (1 + \sigma^2 (\mu_k^i)^T (\Delta_k)^{-1} \mu_k^i)^{-1}.
\]

We consider aggregated data on a disease or mortality outcome are available on the \(K\) populations corresponding to the total number of disease cases (note that the total number of disease cases is aggregated data that should be easy to obtain from governmental agencies) and the total number of individuals at risk \(n_k\) during the study period. For model (7.3), we define

\[
\nu_k^A = (n_k)^{-1} \left( \sum_{i=1}^{n_k} Y_{ki} \right) \quad \text{and} \quad \hat{\mu}_k^A = E(\nu_k^A) = E(E(\nu_k^A | h_k)) = e^{x_{k}^T \alpha},
\]

Then the aggregated-data estimating equation \textsuperscript{127,128} is

\[
\sum_{k=1}^{K} (\hat{D}_k^A)^T (\hat{V}_k^A)^{-1} (\hat{\nu}_k^A - \hat{\mu}_k^A) = 0, \quad (7.5)
\]

where \(\hat{D}_k^A = \hat{\partial} \hat{\mu}_k^A / \hat{\partial} \alpha^T\) and \(\hat{V}_k^A = \sigma^2 \{ (\hat{\mu}_k^A)^2 - \hat{\phi}_k (n_k)^{-1} \} + (\hat{\mu}_k^A - \hat{\phi}_k) (n_k)^{-1}\) with \(\hat{\phi}_k = e_m \{ e^{x_i^T \alpha} \}. \) Note that we don’t observe the individual covariates of every individual in the population, except those in the sample. Prentice and Sheppard,
therefore, estimated the average values $\mu_k^{\Lambda} = \varepsilon_{n_k} \{e^{x_k^T} \alpha\}$ and $\phi_k = \varepsilon_{n_k} \{e^{2x_k^T} \alpha\}$ with the average values in the sample, $\hat{\mu}_k^{\Lambda}$ and $\hat{\phi}_k$, respectively.

In each model, statistical inference on $\alpha$ can generally be based on the asymptotic normality of $\hat{\alpha}$ whose variance can be estimated consistently by the robust-sandwich variance estimator$^{128,34,135}$. For the IRM, we can compute the information matrix as

$$I^i_\alpha = \sum_{k=1}^K (D_k^i)^T (V_k^i)^{-1} D_k^i$$

and the robust-sandwich estimator by

$$\left( I^i_\alpha \right)^{-1} \left[ \sum_{k=1}^K (D_k^i)^T (V_k^i)^{-1} (Y_k^i - \mu_k^i)(Y_k^i - \mu_k^i)^T (V_k^i)^{-1} D_k^i \right] \left( I^i_\alpha \right)^{-1}$$

with all quantities evaluated at $\hat{\alpha}^T$. On the other hand, the ARM has information matrix define as

$$I^i_\alpha = \sum_{k=1}^K (\hat{D}_k^i)^T (\hat{V}_k^i)^{-1} \hat{D}_k^i$$

and robust-sandwich estimator by

$$\left( I^i_\alpha \right)^{-1} \left[ \sum_{k=1}^K (\hat{D}_k^i)^T (\hat{V}_k^i)^{-1} (\hat{Y}_k^i - \hat{\mu}_k^i)(\hat{Y}_k^i - \hat{\mu}_k^i)^T (\hat{V}_k^i)^{-1} \hat{D}_k^i \right] \left( I^i_\alpha \right)^{-1}$$

with all quantities evaluated at $\hat{\alpha}^T$.

The estimation procedure is completed by inserting a $K^{1/2}$-consistent estimator for $\sigma^2$. Such estimators are given by a moment estimators defined for the IRM and ARM$^{127,128}$, respectively:
\[
(\hat{\sigma}^2)_i = \frac{1}{K} \sum_{k=1}^{K} \left[ \left( \varepsilon_{m_k} \{Y_k\} (\varepsilon_{m_k} \{Y_k\} m_k - 2\hat{\mu}_k m_k - 1) + 2\varepsilon_{m_k} \{\mu_k Y_k\} \right) \left[ (\hat{\mu}_k^A)^2 m_k - \hat{\phi}_k \right]^{-1} + 1 \right]
\]

\[
(\hat{\sigma}^2)_A = \frac{1}{K} \sum_{k=1}^{K} \left[ (Y_k^A - \hat{\mu}_k^A)^2 - (\hat{\mu}_k^A - \hat{\phi}_k)(n_k)^{-1} \right] \left[ (\hat{\mu}_k^A)^2 - \hat{\phi}_k (n_k)^{-1} \right]^{-1}.
\]

The estimating equations can be solved through the Newton-Raphson procedures (see R.105 program in appendix A.3).
CHAPTER 8

Geographical regression extension: an integrated analysis of individual and aggregated health outcomes

"Don’t be a novelist, be a statistician, much more scope for the imagination”.

Darell Huff and Mel Calman

8.1 Introduction.

In chapter 7 we considered individual- and aggregated-data analyses of disease-exposure on K populations. As we described, individual- and aggregated-data analyses approaches are useful depending on where the exposure variation exists. However, in epidemiological studies we usually consider two or more covariates (exposures and confounding variables) that can have different types of variations, i.e., we can have covariates with high within-population variability and others with high between-population variability. In these cases, an individual-data analysis approach can perform poorly on the estimation of the covariates with high between-population variability, while an aggregated-data analysis approach can perform poorly on the estimation of the covariates with high within-population variability. In addition, if we have an exposure covariable and a confounding variable that have different within- and between-population variabilities, the individual-data analysis can perform poorly in estimating the exposure effect even if the exposure is subject to a high within-population variability.
This is due to the influence of the confounding covariate, which is by definition related to the exposure of interest and can have high between-population variability. Similarly, the aggregated-data analysis approach could perform poorly in estimating exposure effects even if the exposure is subject to high between-population variability, due to the influence of the confounding variable with high within-population variability.

In this chapter, we consider a new analytical framework that is an integrated data approach based on combining the individual- and aggregated-data analyses, presented in chapter 7. This method uses an estimating equation approach following the original papers of Prentice and Sheppard. The proposed analysis utilizes strengths of both individual- and aggregated-health data analysis approaches in the estimation of the exposure effect of interest, depending on which of the exposure variations (within- vs. between-population) dominates. As we pointed out above, this approach can be useful in epidemiological studies where we include exposure and confounding variables that can have different source of within and between-population variability. For example, in the study of the aetiology of bladder cancer we can jointly include variables where the within-population variability is higher than the between-population variation, such as smoking status, and variables where the between-population variation can be higher than the within-population, such as chlorinated drinking water.

The utilization of both types of data has been proposed under the fully Bayesian framework by Jackson et al. Our proposal follows the same basic concept of Jackson et al., but applies it under the estimating equation approach that Prentice and Sheppard proposed originally.

In Section 8.2, we explain the study design and data structure of the proposed analytical framework. Section 8.3, describe the combination of the individual- and aggregated-data random effects models, a “population-based estimating equation” (PBEE) approach. Section 8.4 describes a simulation study that illustrates advantages of the PBEE over individual- and aggregated-data analyses presented in chapter 7. Finally, Section 8.5 contains discussion.

**8.2 Study design.**
We consider a study design in which 1) aggregated data on a disease or mortality outcome are available on $K$ populations with population sizes $n_k$’s ($k=1,…,K$), and 2) individual data for a random sample of $m_k$ individuals ($m_k \leq n_k$) from the $k$-th population are collected. We denote the disease outcome variable of the $i$-th individual in the $k$-th population as $Y_{ki}$ and a vector of covariates associated as $z_{ki}$. The variable $Y_{ki}$ takes a value of one if the outcome of interest (disease/death) occurs on the individual within the defined study follow-up period, and zero otherwise. In each population’s aggregate data, we have the total number of disease cases $\sum_i Y_{ki}$ and the total number of individuals at risk $n_k$ during the study period, and, possibly, a vector of population-level covariates $Z_k$. These aggregated data are often available and published periodically from governmental agencies. A diagram of the data structure is given in Figure 8.1.

**Figure 8.1** Diagram of the data structure.
8.3 Relative rate inference based on population-based estimating equations.

To utilize the entire data for parameter estimation under the study design of Figure 8.1, we propose to combine estimating equations for the individual- and aggregated-data analyses into one equation. Using $Y_k = (Y_{k1}, ..., Y_{km_k})^T$ and $\mu_k = (\mu_{k1}, ..., \mu_{km_k})^T$ with $\mu_{ki} = E(Y_{ki}) = E(E(Y_{ki} | h_k)) = e^{x_i^T \alpha}$, the individual-data estimating equation is defined as (7.4.), i.e.

$$K_k \sum_{i=1}^{K} (D_k^I)^T (V_k^I)^{-1} (Y_k^I - \mu_k^I) = 0,$$

where $D_k^I = \partial \mu_k^I / \partial \alpha^T$, $V_k^I = \Delta_k + \sigma^2 \mu_k^I (\mu_k^I)^T$ with $\Delta_k = \text{diag} [\mu_{k1} \{1 - (1 + \sigma^2) \mu_{k1} \}]$.

The inverse of the variance-covariance matrix $(V_k^I)^{-1}$ can be computed by

$$(V_k^I)^{-1} = (\Delta_k)^{-1} - \sigma^2 (\Delta_k)^{-1} \mu_k^I (\mu_k^I)^T (\Delta_k)^{-1} \{1 + \sigma^2 (\mu_k^I)^T (\Delta_k)^{-1} \mu_k^I \}^{-1}.$$

For the aggregated part we exclude the individual data from the aggregated data in each population, that is, we now define $\bar{Y}_k^A$ as

$$\bar{Y}_k^A = (n_k - m_k)^{-1} (\sum_{i=1}^{n_k} Y_{ki} - \sum_{i=1}^{m_k} Y_{ki})$$

and $\hat{\mu}_k^A = E(\bar{Y}_k^A) = E(E(\bar{Y}_k^A | h_k)) = \varepsilon_{m_k} \{e^{x_i^T \alpha}\}$.

The aggregated-data estimating equation is

$$K_k \sum_{k=1}^{K} (\hat{D}_k^A)^T (\hat{V}_k^A)^{-1} (\hat{Y}_k^A - \hat{\mu}_k^A) = 0,$$

where $\hat{D}_k^A = \partial \hat{\mu}_k^A / \partial \alpha^T$ and $\hat{V}_k^A = \sigma^2 \{ (\hat{\mu}_k^A)^2 - \hat{\phi}_k (n_k - m_k)^{-1} \} + (\hat{\mu}_k^A - \hat{\phi}_k) (n_k - m_k)^{-1}$ with $\hat{\phi}_k = \varepsilon_{m_k} \{e^{x_i^T \alpha}\}$ (127,128) (see appendix A.4 for demonstration). Note that we don’t observe the individual covariates of every individual in the population, except those in
the sample. We, therefore, estimate the average values \( \mu_k^A = e_{n_k-m_k} \{e^{x'\alpha} \} \) and \( \phi_k = e_{n_k-m_k} \{e^{x'\alpha} \} \) with the average values in the sample, \( \hat{\mu}_k^A \) and \( \hat{\phi}_k \), respectively.

The two estimating equations above can be combined to utilize both the individual and aggregate components of the entire data:

\[
\sum_{k=1}^{K} \left( \begin{array}{c}
    D_k^T \\
    \hat{D}_k^A
  \end{array} \right) \left( \begin{array}{cc}
    V_k^I & 0 \\
    0 & \hat{V}_k^A
  \end{array} \right)^{-1} \left( \begin{array}{c}
    Y_k^I - \mu_k^I \\
    \bar{Y}_k^A - \bar{\mu}_k^A
  \end{array} \right) = 0.
\]

Note that we are proposing a simple addition of the two estimating equations. The combined estimating equation is a slight deviation from the optimal linear estimating function\(^{138}\) of the form,

\[
(\partial E[Y]/\partial \alpha^T) (\text{Var}[Y])^{-1} (Y - E[Y]).
\]

Specifically, \( \left( \begin{array}{cc}
    V_k^I & 0 \\
    0 & \hat{V}_k^A
  \end{array} \right) \) is not the variance-covariance matrix of \( \left( \begin{array}{c}
    Y_k^I - \mu_k^I \\
    \bar{Y}_k^A - \bar{\mu}_k^A
  \end{array} \right) \): \( (Y_k^I - \mu_k^I) \) and \( (\bar{Y}_k^A - \bar{\mu}_k^A) \) are correlated and \( \bar{\mu}_k^A \) has a sampling variation that is unaccounted for in \( \hat{V}_k^A \). As these second-order assumptions are difficult to verify, our proposal is to keep the “weights” of the combined estimating function to correspond to a simple sum of the two estimating equations, and use a robust-sandwich variance estimator\(^{139}\) of \( \hat{\alpha} \) that reflects empirical second-order characteristics of \( \left( \begin{array}{c}
    Y_k^I - \mu_k^I \\
    \bar{Y}_k^A - \bar{\mu}_k^A
  \end{array} \right) \).

This is in the spirit of Prentice and Sheppard\(^{127,128}\) and Liang and Zeger\(^{34,135}\) in their use of a robust-sandwich variance estimator of mean parameters.

Statistical inference on \( \alpha \) can generally be based on the asymptotic normality of \( \hat{\alpha} \) whose variance can be estimated consistently by the following robust-sandwich variance estimator:
\[(I_{\alpha})^{-1} \left[ \sum_{k=1}^{K} \begin{pmatrix} D_k^l \end{pmatrix}^T \begin{pmatrix} V_k^l & 0 \\ 0 & \hat{V}_k^A \end{pmatrix}^{-1} \begin{pmatrix} Y_k^l - \mu_k^l \\ \bar{Y}_k^A - \hat{\mu}_k^A \end{pmatrix}^T \begin{pmatrix} V_k^l & 0 \\ 0 & \hat{V}_k^A \end{pmatrix}^{-1} \begin{pmatrix} D_k^l \end{pmatrix} \right] (I_{\alpha})^{-1} \]

where 

\[I_{\alpha} = \sum_{k=1}^{K} \begin{pmatrix} D_k^l \end{pmatrix}^T \begin{pmatrix} V_k^l & 0 \\ 0 & \hat{V}_k^A \end{pmatrix}^{-1} \begin{pmatrix} D_k^l \end{pmatrix} \]

with all quantities evaluated at \( \hat{\alpha} \). As Prentice and Sheppard\(^{127,128} \), the estimation procedure is completed by inserting a \( K^{1/2} \)-consistent estimator for \( \sigma^2 \). Such estimators are given by a moment estimators defined for the individual and aggregated components of the estimating equation, respectively (see appendix A.4 for demonstration):

\[(\hat{\sigma}^2)_i = \frac{1}{K} \sum_{k=1}^{K} \left[ \left( \varepsilon_{m_k} \{Y_k\}(\varepsilon_{m_k} \{Y_k\} m_k - 2\hat{\mu}_k m_k - 1) + 2\varepsilon_{m_k} \{\mu_k Y_k\} \right) \left( (\hat{\mu}_k^A)^2 m_k - \hat{\phi}_k \right)^{-1} + 1 \right] \]

\[(\hat{\sigma}^2)_A = \frac{1}{K} \sum_{k=1}^{K} \left[ (\bar{Y}_k^A - \hat{\mu}_k^A)^2 - (\hat{\mu}_k^A - \hat{\phi}_k)(n_k - m_k)^{-1} \right] \left( (\hat{\mu}_k^A)^2 - \hat{\phi}_k (n_k - m_k)^{-1} \right)^{-1}. \]

We do not unify the two estimators of \( \sigma^2 \) in line with the idea of combining two estimating equations into one. The estimating equation can be solved through the Newton-Raphson procedures (See R program in appendix A.3).

### 8.4 Simulation design and efficiency comparison.

A simulation study was conducted to compare the inferential performance of three approaches (IRM, ARM, PBEE) described in chapters 7 and section 8.2. We considered four different sample-size scenarios depending on the number of populations, \( K \), and the sample size in each population, \( m_k \): \((K, m_k) = (100,100), (100,50), (50,100), \) and \((50,50)\). These different scenarios can take place in real small-area geographical health studies. The population size \( n_k \) in each population was fixed at 2,000. In each scenario, we considered the following simulation similar to that of Prentice and Sheppard\(^{127,128} \). Two covariates, denoted as \( z_{k1} \) and \( z_{k2} \), were generated, where \( z_{k1} \) represents exposure of interest and \( z_{k2} \) can represent a confounding factor. For the \( i \)-th person in the \( k \)-th population, we generate individual covariate values \((z_{k1i}, z_{k2i})\) from a bivariate normal
distribution with population mean \((z_{k1}, z_{k2})\) and variance-covariance matrix
\[
\begin{pmatrix}
\sigma^2_w & \theta_1 \sigma_w \\
\theta_1 \sigma_w & 1
\end{pmatrix},
\]
where the population mean \((z_{k1}, z_{k2})\) is also a bivariate normal random vector with mean \((0, 0)\) and variance-covariance
\[
\begin{pmatrix}
1 & \theta_p \\
\theta_p & 1
\end{pmatrix}.
\]
To consider a range of exposure variance in within- and between-populations, the within-population variance \(\sigma^2_w\) of exposure, were set at 0.25, 0.5, 1, 2, 4 or 16 but the between-population variance of exposure was fixed at 1. This covers a wide range of the within- vs. between-variance ratio of exposure from 0.25/1 where the between-population variance dominates the within-population variance, to 16/1 where the within-population variance dominates the between-population variance. We considered \((\theta_1, \theta_p) = (0,0)\) for no confounding case (NCC) and \((0.3, 0.3)\) for a simple confounding case (SCC) by \(z_{k2}\). In the NCC and SCC cases, we changed the within- versus between-population ratio and considered that the additional covariate, that in the SCC is a confounding factor, had the same within- and between-population variance. We considered that \((\theta_1, \theta_p) = (0.3, 0.3)\) were reasonable values to analyze the influence of the confounding factor. We also considered an extended confounding case (ECC) with \((\theta_1, \theta_p) = (0.3,0.3)\), where the within- versus between-population variance ratio for the two covariates, \(z_{k1}\) and \(z_{k2}\), interchangeable with respect to their roles as an exposure or as a confounding factor, were above and below one, respectively, i.e., \(z_{k1}\)’s within- and between-population variances are 0.85 and 3.4, respectively, and \(z_{k2}\)’s within- and between-population variances are 4 and 0.25, respectively. In this way, the two covariates have the same total variance, but \(z_{k1}\) has the variance ratio of 0.25 indicating that the between-population variance dominates this covariate’s variability and \(z_{k2}\) has the variance ratio of 16 indicating that the within-population variance dominates.

With the NCC and SCC, we can study situations where an exposure covariate can have different within- and between-population variabilities, and, in addition, we can have, respectively, a no-confounding and a confounding covariate which have the same within- and between-population variance. In this way, we can evaluate if 1) the IRM performs well when the within-population variability is high for the exposure of interest.
and poorly when the between-population variability is high, 2) the ARM performs well when the between-population variability is high and poorly when the within-population variability is high, and 3) if the PBEE approach performs well in both situations.

With the ECC, we can study the situation where we have an exposure covariate with different within- and between-population variabilities with respect to a confounding variable. In this way, we can evaluate if 1) the IRM performs poorly in estimating the exposure effect in spite of high within-population variability, due to the influence of the confounding factor which may not be estimated well because of its high between-population variability, 2) the ARM performs poorly in estimating the exposure effect in spite of high between-population variability, due to the influence of confounding variable which may not be estimated well because of its high within-population variability and 3) the PBEE approach performs well in the two situations described.

Population-specific frailties, $h_k$, were generated as independent realized values from a gamma distribution with mean 1 and variance 0.05. The disease event indicator, $Y_{ki}$, was generated as a Bernoulli random variable with probability $h_k \exp(\gamma_0 + \beta_1 z_{ki1} + \beta_2 z_{ki2})$ where $\gamma_0 = -3$, $\beta_1 = 0.2$, and $\beta_2 = 0.2$. The 0.2 value represents a relative risk value of approximately 1.2 that reflects a positive association with the disease outcome for the exposure and confounding factors. A total of 1,000 simulation runs were carried out in each of the different parameter sets. The simulation was programmed in the free software R\textsuperscript{105} (see appendix A.3 for a detailed description of the steps followed).

Tables 8.1 and 8.2 present the bias and coverage of the 95% confidence interval for the 1,000 simulation runs for NCC and SCC. These results were presented in all the four $(K, m_k)$ combinations for the three approaches: individual random effects model (IRM), aggregated random effects model (ARM) and the PBEE approach. For NCC (Table 8.1), when the between-population variance is larger than the within-population variance, the ARM model generally presents lower bias than the IRM model, and when the within-population variance is larger, the IRM generally presents lower bias than the ARM model. The PBEE approach presents either the lowest value, or close to the
Table 8.1 Bias and 95% confidence interval coverage for the individual random effects model (IRM), aggregated random effects model (ARM) and population-based estimating equation approach (PBEE) for the no confounding case (NCC), in the different within- and between-population variance ratios for the four scenarios \((K, m_k) = (100,100), (100,50), (50,100), \) and \((50,50)\).

| Number of Populations | Variance Ratio | Survey sample size in k-th population | 100 | 50 | 50%
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>% Bias</td>
<td>95% Interval coverage</td>
<td>% Bias</td>
<td>95% Interval coverage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IRM</td>
<td>ARM</td>
<td>PBEE</td>
<td>IRM</td>
</tr>
<tr>
<td>0.25</td>
<td>0.5</td>
<td>0.07</td>
<td>-0.07</td>
<td>0.00</td>
<td>92</td>
</tr>
<tr>
<td>1</td>
<td>0.74</td>
<td>-0.56</td>
<td>-0.08</td>
<td>0.94</td>
<td>92</td>
</tr>
<tr>
<td>2</td>
<td>0.90</td>
<td>-2.22</td>
<td>-0.56</td>
<td>94</td>
<td>93</td>
</tr>
<tr>
<td>4</td>
<td>0.99</td>
<td>-4.63</td>
<td>-0.59</td>
<td>94</td>
<td>91</td>
</tr>
<tr>
<td>8</td>
<td>0.39</td>
<td>-8.98</td>
<td>-1.01</td>
<td>92</td>
<td>87</td>
</tr>
<tr>
<td>16</td>
<td>-2.79</td>
<td>-19.16</td>
<td>-3.47</td>
<td>90</td>
<td>56</td>
</tr>
<tr>
<td>0.5</td>
<td>0.85</td>
<td>0.32</td>
<td>0.60</td>
<td>91</td>
<td>93</td>
</tr>
<tr>
<td>1</td>
<td>-1.49</td>
<td>-1.13</td>
<td>-1.24</td>
<td>93</td>
<td>92</td>
</tr>
<tr>
<td>2</td>
<td>-0.55</td>
<td>-1.56</td>
<td>-1.14</td>
<td>93</td>
<td>92</td>
</tr>
<tr>
<td>4</td>
<td>0.62</td>
<td>-2.50</td>
<td>-0.92</td>
<td>93</td>
<td>92</td>
</tr>
<tr>
<td>8</td>
<td>1.25</td>
<td>-4.71</td>
<td>-0.60</td>
<td>92</td>
<td>91</td>
</tr>
<tr>
<td>16</td>
<td>1.14</td>
<td>-9.44</td>
<td>-0.45</td>
<td>92</td>
<td>88</td>
</tr>
<tr>
<td></td>
<td>-2.32</td>
<td>-19.20</td>
<td>-3.04</td>
<td>91</td>
<td>74</td>
</tr>
</tbody>
</table>

Table 8.2 Bias and 95% confidence interval coverage for the individual random effects model (IRM), aggregated random effects model (ARM) and population-based estimating equation approach (PBEE) for the simple confounding case (SCC), in the different within- and between-population variance ratios for the four scenarios \((K, m_k) = (100,100), (100,50), (50,100), \) and \((50,50)\).

| Number of Populations | Variance Ratio | Survey sample size in k-th population | 100 | 50 | 50%
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>% Bias</td>
<td>95% Interval coverage</td>
<td>% Bias</td>
<td>95% Interval coverage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IRM</td>
<td>ARM</td>
<td>PBEE</td>
<td>IRM</td>
</tr>
<tr>
<td>0.25</td>
<td>0.5</td>
<td>0.26</td>
<td>0.29</td>
<td>0.59</td>
<td>92</td>
</tr>
<tr>
<td>1</td>
<td>-0.30</td>
<td>-0.84</td>
<td>-0.48</td>
<td>0.33</td>
<td>93</td>
</tr>
<tr>
<td>2</td>
<td>-0.33</td>
<td>-1.43</td>
<td>-0.83</td>
<td>0.33</td>
<td>93</td>
</tr>
<tr>
<td>4</td>
<td>-0.08</td>
<td>-2.87</td>
<td>-1.12</td>
<td>0.68</td>
<td>93</td>
</tr>
<tr>
<td>8</td>
<td>-0.29</td>
<td>-11.44</td>
<td>-1.01</td>
<td>0.29</td>
<td>94</td>
</tr>
<tr>
<td>16</td>
<td>-5.01</td>
<td>-22.89</td>
<td>-5.34</td>
<td>84</td>
<td>45</td>
</tr>
<tr>
<td>0.5</td>
<td>0.71</td>
<td>0.28</td>
<td>0.48</td>
<td>0.71</td>
<td>91</td>
</tr>
<tr>
<td>1</td>
<td>1.34</td>
<td>-0.13</td>
<td>0.11</td>
<td>1.34</td>
<td>90</td>
</tr>
<tr>
<td>2</td>
<td>1.74</td>
<td>-0.83</td>
<td>0.19</td>
<td>1.74</td>
<td>90</td>
</tr>
<tr>
<td>4</td>
<td>1.11</td>
<td>-2.38</td>
<td>-0.14</td>
<td>1.11</td>
<td>92</td>
</tr>
<tr>
<td>8</td>
<td>1.51</td>
<td>-5.08</td>
<td>-0.09</td>
<td>1.51</td>
<td>92</td>
</tr>
<tr>
<td>16</td>
<td>0.76</td>
<td>-10.91</td>
<td>-0.75</td>
<td>0.76</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>-4.55</td>
<td>-22.54</td>
<td>-5.03</td>
<td>86</td>
<td>68</td>
</tr>
</tbody>
</table>
lowest value, of bias in the three approaches, regardless of the within- and between-variances. The results for the confidence coverage interval are similar for the three models when the variance ratio is not large: they are all slightly lower than the 95% coverage. However, when the within-variance dominates, the ARM’s coverage probability goes low due to the large bias in estimating the parameter: the PBEE approach is affected by the same problem but to a much lesser degree. Similar results and patterns are observed for SCC (Table 8.2).

Figure 8.2 *Mean square error for the individual random effects model (IRM), aggregated random effects model (ARM) and population-based estimating equation approach (PBEE) for the no confounding case (NCC), in the different within- and between-population variance ratios for the four scenarios (K, m_k) = (100,100), (100,50), (50,100), and (50,50).*
Figures 8.2 and 8.3 show mean squared errors of parameter estimation for NCC and SCC, respectively. In all four (K, m_k) combinations for the NCC and SCC, the mean squared error of the IRM decreases, and that of ARM increases as the ratio of within- to between-population exposure variance increases. However, the PBEE approach consistently provides the smallest (or close to the smallest) mean squared errors among the three methods in all the scenarios considered.

**Figure 8.3** Mean square error for the individual random effects model (IRM), aggregated random effects model (ARM) and population-based estimating equation approach (PBEE) for the simple confounding case (SCC), in the different within- and between-population variance ratios for the four scenarios (K, m_k) = (100,100), (100,50), (50,100), and (50,50).
Table 8.3 compares the estimation performance for ECC. The bias is small in both $\beta_1$ and $\beta_2$ estimation by the three approaches except for ARM’s $\beta_2$ estimation subject to the large within-vs. between-variation of $z_{k2}$. In terms of the mean squared error, the ARM results in smaller errors than the IRM for the covariate with 0.25 variance ratio and, while the IRM provides smaller errors than the ARM for the covariate with 16 variance ratio. However, the PBEE approach performs well for the two covariates variance ratios providing the best (or close to the best) results in all the different scenarios we considered.

**Table 8.3** Bias and mean square error (mse) for the individual random effects model (IRM), aggregated random effects model (ARM) and population-based estimating equation approach (PBEE) for the extended confounding case (ECC), in the different within- and between-population variance ratios for the four scenarios $(K, m_k) = (100,100)$, $(100,50)$, $(50,100)$, and $(50,50)$.

<table>
<thead>
<tr>
<th>Number of Populations</th>
<th>Parameter</th>
<th>100 % Bias</th>
<th>Mse x 10^{-3}</th>
<th>50 % Bias</th>
<th>Mse x 10^{-3}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IRM</td>
<td>ARM</td>
<td>PBEE</td>
<td>IRM</td>
<td>ARM</td>
</tr>
<tr>
<td>100</td>
<td>$\beta_1$</td>
<td>0.03</td>
<td>0.86</td>
<td>-0.09</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>$\beta_2$</td>
<td>0.27</td>
<td>-18.04</td>
<td>-1.51</td>
<td>0.65</td>
</tr>
<tr>
<td>50</td>
<td>$\beta_1$</td>
<td>-0.04</td>
<td>0.83</td>
<td>0.12</td>
<td>2.01</td>
</tr>
<tr>
<td></td>
<td>$\beta_2$</td>
<td>0.72</td>
<td>-17.59</td>
<td>-1.43</td>
<td>1.29</td>
</tr>
</tbody>
</table>

### 8.5 Discussion.

This chapter, considered an “integrated” data analysis method, valuable for epidemiological purposes, that combines all the available information in the health outcomes and exposure variables at the different levels of data organization. This analysis performed with the proposed PBEE method presents a powerful analytical framework that takes into account both within- and between-population exposure variation and combines the strengths of both individual- and aggregated-data. It is applicable with the same study design and data structure (Figure 8.1) as the aggregated data analysis and without knowledge of which of the exposure variations (within- vs.
between-population) dominates. In addition, although we may have knowledge of which variations dominates on each variable, two or more exposure variables of interest do not necessarily have the same type of variations. As we have shown by ECC simulations, the PBEE approach will be particularly more advantageous over an individual- or aggregated-data analyses in such cases.

While the PBEE approach shares the same basic concept with the fully Bayesian approach of Jackson et al.,\(^{137}\) in that it utilizes both individual and aggregated data in epidemiological analyses, its estimating equation approach following Prentice and Sheppard\(^{127,128}\) makes mean-parameter inference robust against misspecification of the second-order characteristics of disease outcomes. In addition, Jackson et al.\(^{137}\) multiplied the likelihood of individual data and that of aggregated data, while we removed the samples in the individual data from the aggregated data. If the sample sizes of individuals (\(m_k\)’s) is appreciable, this difference may be important. As a future research topic, a more detailed study can be pursued to evaluate the advantages and disadvantages of the PBEE approach versus the fully Bayesian approach proposed by Jackson et al.

At least, some extensions or modifications are possible for the PBEE approach. We can study multiple imputation techniques to estimate the covariate information in the aggregated part of the PBEE approach\(^{140}\) and adapt it epidemiological designs other than cohort studies, such as case-control data. In addition, it may be of interest to study the efficiency of the estimates obtained with the true variance-covariance matrix vs. with other different choices. Another important point is to apply the PBEE approach to an example with real data. Currently, we don’t have real data yet because the process to obtain individual data is slow due to legal steps that we have to be followed due to confidentiality reasons, as we explained in Chapter 1.

The study design/data structure considered is currently not a common study design in epidemiology, but it offers certain advantages discussed in this chapter and in a previous article by Jackson et al.\(^{137}\). The gain in the parameters estimates can be achieved easily using aggregated mortality or disease data available from governmental agencies. For this and other results shown in this chapter, we recommended the
“integrated” design/data structure with the PBEE approach as a new analytical framework that can be considered in future epidemiological studies.
APPENDICES

A.1 Publications derived from the thesis.

A.1.1 Books.


A.1.2 Book chapters.


A.1.3 Scientific articles.


A.1.4 Scientific conferences.


A.2 General SAS program part 1 (chapter 6).

*GENERAL PROGRAM CATALONIA (RELATIVE RISK AND TREND ESTIMATES) *
*This is a general program, for some causes it is necessary not* 
*include age groups where there are not deaths * 
******************************************************************************

Data Cat2;
  INFILE 'C:\jmiguel\CARPETA ATLAS DE CATALUÑA\DATOS\data8498_cat.txt' DLM='09'x;
  input zone age sex year cause observed population;
  age1=0; age2=0; age3=0; age4=0; age5=0; age6=0; age7=0; age8=0; age9=0;
  age10=0; age11=0; age12=0; age13=0; age14=0; age15=0; age16=0; age17=0; age18=0;
  IF age=1 then age1=1; IF age=2 then age2=1; IF age=3 then age3=1;
  IF age=4 then age4=1; IF age=5 then age5=1; IF age=6 then age6=1;
  IF age=7 then age7=1; IF age=8 then age8=1; IF age=9 then age9=1;
  IF age=10 then age10=1; IF age=11 then age11=1; IF age=12 then age12=1;
  IF age=13 then age13=1; IF age=14 then age14=1; IF age=15 then age15=1;
  IF age=16 then age16=1; IF age=17 then age17=1; IF age=18 then age18=1;
  IF year=8486 then yearr=1; IF year=8789 then yearr=2; IF year=9092 then yearr=3;
  IF year=9395 then yearr=4; IF year=9698 then yearr=5;
run;

proc freq data=Cat2;
weight observed;
tables sex*causer*yearr/nocol norow nocum nopercent;
run;

proc sort data=Cat2;
by zone age sex causer;
run;

data Cat2_Agr1;
set Cat2;
by zone age sex causer;
retain observado;
if first.zone or first.age or first.sex or first.causer then observado=observed;
else observado=(observado+observed);
if not(last.zone or last.age or last.sex or last.causer)then delete;
keep zone age sex causer observado;
run;

data Cat2_Agr2;
set Cat2;
by zone age sex causer;
retain poblacion;
if first.zone or first.age or first.sex or first.causer then poblacion=population;
else poblacion=(poblacion+population);
if not(last.zone or last.age or last.sex or last.causer) then delete;
keep zone age sex causer poblacion;
run;

proc sort data=Cat2_Agr1;
by zone causer age sex;
run;

proc sort data=Cat2_Agr2;
by zone causer age sex;
run;

data final_Cat2;
MERGE Cat2_Agr1 Cat2_Agr2;
by zone causer age sex;
run;

data final_Cat3;
set final_Cat2;
lpy=log(poblacion);
run;

%macro DATA_EXPM;
%do i=1 %to 18;

177
Data Catm_&i;
set final_Cat3;
if cause=&i and sex=1;
run;
%MEND;
%DATA_EXPM;

*GEE approach to obtain reference rates internally;
%macro DATAGEEM;
%do i=1 %to 18;
PROC GENMOD data=Catm_&i;
class zone age;
model observado=age /d=poisson offset=lpy;
repeated subject=zone/type=exch;
ods output GEEmpPEst=gee&i;
run;
%end;
%MEND
%DATA_EXPM;

*Reference rates;
%macro DATA_EXPM2;
%do i=1 %to 18;
Data gee2_&i;
set gee&i;
dum=1;
run;
%end;
%MEND
%DATA_EXPM2;

%macro DATA_EXPM3;
%do i=1 %to 18;
Data gee3_&i;
set gee2_&i;
by dum;
retain valor;
if first.dum then valor=estimate;
rate=exp(valor+estimate);
run;
Data gee4_&i;
set gee3_&i;
cause=&i;
if level1 <> ''; 
age=level1;
keep cause age estimate valor rate;
run;
%end;
%MEND
%DATA_EXPM3;

%macro DATA_EXPM4;
%do i=2 %to 18;
PROC APPEND BASE=gee4_1 DATA=gee4_&i;
RUN;
%end;
%MEND
%DATA_EXPM4;

data geeemale1;
set gee4_1;
agen=age*
1;
causer=cause;
keep causer agen rate;
run;

data Cat3;
set Cat2;
if sex=1;
agen=age;
keep zone causer yearr agen observed population;
run;
proc sort data=geemale1;
by causer agen;
run;
proc sort data=Cat3;
by causer agen;
run;

data fmale1;
MERGE Cat3 geemale1;
by causer agen;
run;

*Expected deaths;
data fmale2;
set fmale1;
expected=population*rate;
run;

*Observed deaths;
proc sort data=fmale2;
by zone yearr causer;
run;
data fmale2_Agr1;
set fmale2;
by zone yearr causer;
retain obs;
if first.zone or first.yearr or first.causer then obs=observed;
else obs=(obs+observed);
if not(last.zone or last.yearr or last.causer) then delete;
keep zone yearr causer obs;
run;
data fmale2_Agr2;
set fmale2;
by zone yearr causer;
retain expec;
if first.zone or first.yearr or first.causer then expec=expected;
else expec=(expec+expected);
if not(last.zone or last.yearr or last.causer) then delete;
keep zone yearr causer expec;
run;

*Merge observed and expected deaths;
proc sort data=fmale2_Agr1;
by zone causer yearr;
run;
proc sort data=fmale2_Agr2;
by zone causer yearr;
run;
data final_male1;
MERGE fmale2_Agr1 fmale2_Agr2;
by zone causer yearr;
run;

%macro DATOS;
%do i=1 %to 18;
Data Datosm_&i;
set final_male1;
if causer=&i;
run;
%end;
%MEND;
%DATOS;

*Empirical Bayes model;
%macro CATALONIAMALE;
%do i=1 %to 18;
PROC NL MIXED DATA=Datosm_&i TECH=TRUREG;
parms log_s=0 log_t=0 beta0=0;
ETA=beta0 + &a*(yearr-3) + &b + &s*(yearr-3);
LAMDA=EXP(ETA);
MEAN=LAMDA*EXPEC;
MODEL OBS ~ POISSON(MEAN);
RANDOM si bi ~ normal ([0,0],
        [exp(2*log_s),0],0,
        [exp(2*log_t),0]);
RUN;
%end;
%MEND;
exp(2*\log_{\tau})
subject=zone OUT=Chmap1_a&i;
run;

Data Chmapaf&i;
set Work.Chmap1_a&i;
RR=exp(Estimate);
keep zone effect estimate probt RR;
run;

Proc sort data=Chmapaf&i;
by zone;
run;

Proc sort data=datosm&_i;
by zone;
run;

data datosmfobs&_i;
set datosm&_i;
by zone;
retain o;
if first.zone then o=obs;
else o=(o+obs);
if not(last.zone) then delete;
keep zone o;
run;

data datosmfexp&_i;
set datosm&_i;
by zone;
retain e;
if first.zone then e=expec;
else e=(e+expec);
if not(last.zone) then delete;
keep zone e;
run;

Proc sort data=datosmfobs&_i;
by zone;
run;
Proc sort data=datosmfexp&_i;
by zone;
run;

data fin_male&_i;
MERGE datosmfobs&_i datosmfexp&_i Chmapaf&i;
by zone;
run;

Data Chmaplarr&_i;
set fin_male&_i;
if effect='bi';
smr=o/e;
run;

Data Chmaplatrend&_i;
set fin_male&_i;
if effect='si';
run;

PROC EXPORT DATA= Work.Chmaplarr&_i
OUTFILE= "C:\jmiguel\CARPETA ATLAS DE CATALUÑA\RESULTADOS\MAP\Resrra&_i..xls"
DBMS=EXCEL2000 REPLACE;
RUN;

PROC EXPORT DATA= Work.Chmaplatrend&_i
OUTFILE="C:\jmiguel\CARPETA ATLAS DE CATALUÑA\RESULTADOS\MAP\Restrenda&_i..xls"
DBMS=EXCEL2000 REPLACE;
RUN;
%end;
%MEND

%CATALONIAMALE;
A.3 R programs part 2 (chapters 7 and 8).

A.3.1 NCC simulation program.

```r
library(MASS)

####################################################
#FUNCTION INDIVIDUAL RANDOM EFFECTS MODEL (farem)#
#FOR COMPUTE THE GRADIENT, HESSIAN AND SIGMA^2  #
####################################################.

farem<-function(betanew,data) {

  # We suppose the next structure in the dataset: Id (Identification of individual).
  # Group (Group number), Y (Individual outcome: 1 death, 0 alive), o (Population's.
  # observed deaths), n (Population at risk), z1 (covariate z1), z2 (covariate z2),
  # ...,zp (covariate Zp).

  Dataprove=data
  K<-dim(Dataprove)[2]
  p<-dim(Dataprove)[2]-5

  for (i in 1:k) ngr[i]<-dim(subset(Dataprove,Dataprove[2]==i))[1]

  gamma0<-betanew[1]
  beta=betanew[-1]

  # Individual outcome.
  Yki<-matrix(,nrow=N,ncol=1)
  Yki[,1]<-Dataprove[,3]

  # Individual mean.
  muki<-matrix(,nrow=N,ncol=1)
  muki[,1]<-exp(gamma0+as.vector(as.matrix(Dataprove[,6:(5+p)])%*%beta))

  # Matrix D for the IRM.
  Dki<-matrix(,nrow=N,ncol=p+1)
  Dki[,1]<-muki[,1]
  for (j in 1:p){
    Dki[,j+1]<-as.numeric(Dataprove[,5+j])*muki[,1]
  }

  # Inverse variance-covariance matrix for the IRM.
  muki2<-matrix(,nrow=N,ncol=1)
  muki2[,1]<-muki[,1]^2

  Yaver<-matrix(,nrow=1,ncol=K)
  muk<-matrix(,nrow=1,ncol=K)
  phik<-matrix(,nrow=K,ncol=1)
  sigma2rek<-matrix(,nrow=K,ncol=1)

  ini<-1
  end<-ngr[1]
  for (i in 1:k) {
    Yaver[1,i]<-sum(Yki[ini:end])/ngr[i]
    muk[1,i]<-sum(muki[ini:end])/ngr[i]
    phik[i,1]<-sum(muki2[ini:end])/ngr[i]
    sigma2rek[i,1]<-max((Yaver[1,i]*Yaver[1,i]*muk[1,i])/(Yaver[1,i]*muk[1,i]+1),
                        (Yaver[1,i]*muk[1,i]*1.1)+2*(t(Dki[,i]*muk[,1]*1.1)))/ngr[i])
    ini<-end
    end<-ngr[i+1]
  }
  sigma2re<-sum(sigma2rek[1:K])/K

  # We compute the expression for one part (transpose(muk)*Inverse(Deltak)*muk).
  sumk<-matrix(,nrow=1,ncol=K)
  ini<-1
  for (i in 1:k) {
    sumk[1,i]<-sum(muki[ini:end]^2)/(muki[ini:end]*muki[ini:end])
    ini<-end+1
  }
  sumk<-sumk[1:K]
  # This will be used for the expression of the gradient.
  H<-matrix(,nrow=K,ncol=K)
  for (i in 1:K) {
    for (j in 1:K) {
      H[i,j]<-sumk[j,i]*sigma2rek[i,1]
    }
  }
}
```

Finally, we define the elements of the inverse of $V$.

```r
Vki <- list(matrix(, nrow=K, ncol=1))
for (i in 1:K) {
  Vki[[i]] <- matrix(0, nrow=ngr[1], ncol=ngr[1])
  for (j in 1:ngr[1]) {
    yy = 1 - (1 + sigma2re * muki[j])
    Vki[[i]][j, j] <- (1 + sigma2re * sumk[i, j]) / (1 + sigma2re * sumk[i, j])
    Vki[[i]][i, i] <- (1/muki[j]) - (sigma2re/(1/muki[j]))
    Vki[[i]] = Vki[[i]] + t(Vki[[i]]) - diag(diag(Vki[[i]]))
  }
}
```

# Gradient, Hessian for IRM#

# Vectors of individual responses for each group. For ngr[4] is NA but we don't use it.
Ykilist <- list(matrix(, nrow=K, ncol=1))
# Vectors of mean for each group.
mukilist <- list(matrix(, nrow=K, ncol=1))
ini = 1
end = ngr[1]
for (i in 1:K) {Ykilist[[i]] <- list(matrix(Dataprove[ini:end, 3], nrow=ngr[i], ncol=1)) mukilist[[i]] <- list(matrix(muki[ini:end], nrow=ngr[i], ncol=1)) ini = end + 1 end = ngr[i + 1] + end}

# Vector difference response and mean.
Ykiminusmuki <- list(matrix(, nrow=K, ncol=1))
for (i in 1:K) {Ykiminusmuki[[i]] <- (Ykilist[[i]] - mukilist[[i]])}

# Matrix Dk for each group.
Dkilist <- list(matrix(, nrow=K, ncol=1))
for (i in 1:K) Dkilist[[i]] <- matrix(, nrow=ngr[i], ncol=p + 1)
ini = 1
end = ngr[1]
for (j in 1:K) {
  for (n in 1:(p + 1)) {Dkilist[[j]][, n] <- Dki[ini:end, n] ini = end + 1 end = ngr[j + 1] + end}
}

# Element KGr
ElementKGr <- list(matrix(, nrow=K, ncol=1))
Grlist <- list(matrix(0, nrow=(p + 1), ncol=1))
for (i in 1:K) {
  ElementKGr[[i]] <- t(Dkilist[[i]]) %*% Vki[[i]] %*% Ykiminusmuki[[i]]
  Grlist[[i]] <- ElementKGr[[i]] + Grlist[[i]]
}
Gr <- matrix(, nrow=(p + 1), ncol=1)
for (i in 1:(p + 1)) {Gr[i, 1] <- Grlist[[i]]

# Element KHs
ElementKHs <- list(matrix(, nrow=K, ncol=1))
Hslist <- list(matrix(0, nrow=(p + 1), ncol=(p + 1)))
for (i in 1:K) {
  ElementKHs[[i]] <- t(Dkilist[[i]]) %*% Vki[[i]] %*% Dkilist[[i]]
  Hslist[[i]] <- ElementKHs[[i]] + Hslist[[i]]
}
Hs <- matrix(, nrow=(p + 1), ncol=(p + 1))
for (i in 1:(p + 1)) {
  for (j in 1:(p + 1)) {Hs[i, j] <- Hslist[[i]][j, j]}}

# We return the Gradient (Gr), Hessian (Hs) and sigma^2(sigma2re).
list(Gradient=Gr, Hessian=Hs, sigma2re=sigma2re)
```

# Function Aggregated Random Effects Model (Fagrem)#

```r
fagrem <- function(betanew, data) {
  # We suppose the next structure in the dataset: Id (Identification of individual),
```
#Group (Group number), Y (Individual outcome: 1 death, 0 alive), O (Population's observed deaths), n (Population at risk), Z1 (covariate Z1), Z2 (covariate Z2), ..., Zp (covariate Zp).

Dataprove=data

#K is the number of groups. We suppose that groups are ordered and they have all the correlatives numbers. For example: 1, 2, 3 and not 1, 3 (There are no number 2).
#N is the number of observations and p is the number of covariates.
N<-dim(Dataprove)[1]
K<-Dataprove[N,2]
p<-dim(Dataprove)[2]-5
ngr<-matrix(nrow=1,ncol=K)
for (i in 1:K) ngr[1,i]<-dim(subset(Dataprove,Dataprove[2]==i))[1]

gamma0<-betanew[1]
beta<-betanew[-1]

#Individual mean (muki).
muki<-matrix(nrow=N,ncol=1)
muki[,1]<-exp(gamma0+as.vector(as.matrix(Dataprove[,6:(5+p)]))%*%beta))

#Individual matrix D.
Dki<-matrix(nrow=N,ncol=p+1)
Dki[,1]<-muki[,1]
for (j in 1:p){
  Dki[,j+1]<-as.numeric(Dataprove[,5+j])*muki[,1]
}

#Variance for the ARM.
muki2<-matrix(nrow=N,ncol=1)
muki2[,1]<-muki[,1]^2

#Outcome for the ARM as defined in Sheppard and Prentice (Biometrics, 1995).
Y<-matrix(nrow=1,ncol=K)
muk<-matrix(nrow=1,ncol=K)
phik<-matrix(nrow=K,ncol=1)

#Matrix D for the ARM.
Dk<-matrix(nrow=K,ncol=p+1)

#First, we compute sigma square.
sigma2amk<-matrix(nrow=K,ncol=1)
in<1
end<-ngr[1]
for (i in 1:K) {
  Y[i,1]<-((Dataprove[ini,4])/(Dataprove[ini,5]))
  muk[i,1]<-sum(muki[ini:end])/ngr[i]
  phik[i,1]<-sum(muki2[ini:end])/ngr[i]
  for (j in 1:p+1) {Dk[i,j]<-sum(Dki[ini:end,j])/ngr[i]
    sigma2amk[i,1]<-max(((Y[i,1]-muk[i,1])^2-(muk[i,1]-phik[i,1])/(Dataprove[ini,5])),100)
  }
}

sigma2am<-sum(sigma2amk[1:K])/K

#Finally, we define the variance.
Vk<-matrix(nrow=1,ncol=K)
in<1
end<-ngr[1]
for (i in 1:K) {Vk[i,1]<-sigma2am*(muk[i,1]^2)-(phik[i,1]/(Dataprove[ini,5]))+(muk[i,1]-phik[i,1])/(1/(Dataprove[ini,5]))
  ini<1
  end<-ngr[i+1]+end}

#Gradient.
Dkt<-t(Dk)
ElementkAGr<-list(matrix(nrow=K,ncol=1))

#Gradient, Hessian for ARM.

#Gradient.
Dkt<-t(Dk)
ElementkAGr<-list(matrix(nrow=K,ncol=1))

#Gradient.
Dkt<-t(Dk)
ElementkAGr<-list(matrix(nrow=K,ncol=1))

#Finally, we define the variance.
Vk<-matrix(nrow=1,ncol=K)
in<1
end<-ngr[1]
for (i in 1:K) {Vk[i,1]<-sigma2am*(muk[i,1]^2)-(phik[i,1]/(Dataprove[ini,5]))+(muk[i,1]-phik[i,1])/(1/(Dataprove[ini,5]))
  ini<1
  end<-ngr[i+1]+end}

#Finally, we define the variance.
Vk<-matrix(nrow=1,ncol=K)
in<1
end<-ngr[1]
for (i in 1:K) {Vk[i,1]<-sigma2am*(muk[i,1]^2)-(phik[i,1]/(Dataprove[ini,5]))+(muk[i,1]-phik[i,1])/(1/(Dataprove[ini,5]))
  ini<1
  end<-ngr[i+1]+end}

#Finally, we define the variance.
Vk<-matrix(nrow=1,ncol=K)
in<1
end<-ngr[1]
for (i in 1:K) {Vk[i,1]<-sigma2am*(muk[i,1]^2)-(phik[i,1]/(Dataprove[ini,5]))+(muk[i,1]-phik[i,1])/(1/(Dataprove[ini,5]))
  ini<1
  end<-ngr[i+1]+end}
Hsa<-matrix(,nrow=(p+1),ncol=(p+1))
for (i in 1:(p+1)) {for (j in 1:(p+1)) {Hsa[i,j]<-Hsalist[[1]][i,j]}}

## We return the Gradient (Gra), Hessian(Hsa) and sigma^2(sigma2am).
list(Gradienta=Gra,Hessiana=Hsa,sigma2am=sigma2am)}

########################################################.
## FUNCTION POPULATION-BASED ESTIMATING EQUATION (fpbm)**.
########################################################.
fpbm<-function(betanew,data) {
  # We suppose the next structure in the dataset:Id (Identification of individual),
  # Group (Group number), Y (Individual outcome: 1 death, 0 alive), O (Population's. 
  # observed deaths), n (Population at risk), Z1 (covariate Z1), Z2 (covariate Z2),
  # ..., Zp (covariate Zp).
  Dataprove=data
  K<-dim(Dataprove)[2]
  K<-Dataprove[N,2]
  p<-dim(Dataprove)[2]-5
  ngr<-matrix(,nrow=1,ncol=K)
  for (i in 1:K) ngr[1,i]<-dim(subset(Dataprove,Dataprove[2]==i))[1]
  gamma0<-betanew[1]
beta<-betanew[-1]

  # Individual outcome.
  Yki<-matrix(,nrow=N,ncol=1)
  Yki[,1]<-Dataprove[,3]
  muki<-matrix(,nrow=N,ncol=1)
  muki[,1]<-exp(gamma0+as.vector(as.matrix(Dataprove[,6:(5+p)])%*%beta))
  # Individual matrix D.
  Dki<-matrix(,nrow=N,ncol=p+1)
  Dki[,1]<-muki[,1]
  for (j in 1:p){
    Dki[,j+1]<-as.numeric(Dataprove[,5+j])*muki[,1]
  }
  # Inverse variance-covariance matrix individual part.
  # First, we compute sigma square for the individual part.
  muki2<-matrix(,nrow=N,ncol=1)
  muki2[,1]<-muki[,1]^2
  Yaver<-matrix(,nrow=1,ncol=K)
  muk<-matrix(,nrow=1,ncol=K)
  phik<-matrix(,nrow=K,ncol=1)
  sigma2rek<-matrix(,nrow=K,ncol=1)
  ini<-1
  end<-ngr[1]
  for (i in 1:K) {
    Yaver[1,i]<-sum(Yki[ini:end])/ngr[i]
    muk[1,i]<-sum(muki[,ini:end])/ngr[i]
    phik[1,i]<-sum(muki2[,ini:end])/ngr[i]
    sigma2rek[i]<-max((Yaver[1,i]*(Yaver[1,i]*ngr[i]-2*ngr[i]*muk[1,i]-1)+2*((t(muki[,ini:end,1])*Yki[,ini:end,1])/ngr[i])(ngr[i]/(muk[1,i]^2-phik[1,i,1])+1,-100))
    ini<--end
    end<-ngr[i+1]+end
  }
  sigma2re<-sum(sigma2rek[1:K])/K
  # We compute the expression for one part (transpose(muk)*Inverse(Deltak)*muk).
  sumk<-matrix(,nrow=1,ncol=K)
  ini<-1
  end<-ngr[1]
for (i in 1:K) {sumk[1,i]<-sum((muki[ini:end]^2)/(muki[ini:end]*(1-
(1+sigma2re)*muki[ini:end])))
ini<-end+1
end<-ngr[i+1]+end}

#Finally we define the elements of the inverse of v.
vki<-list(matrix(,nrow=K,ncol=1))
for (j in 1:K){
  vki[j]<-matrix(0,nrow=ngr[j],ncol=ngn[1])
  for (i in 1:ngr[j]) {
    y<-1-(1+sigma2re)*muki[i]
    vki[[j]][i,]<-(-sigma2re*(1/yy)*(1/(1-(1+sigma2re)*muki[i])))
    vki[[j]][i,i]<-(-sigma2re*(1/yy)^2)*(1/(1+sigma2re*sumk[1,j]))
  }
  vki[[j]]=vki[[j]]+t(vki[[j]])-diag(diag(vki[[j]]))
}

#Vectors of individual responses for each group. For ngr[4] is NA but we don't use it.
ykilist<-list(matrix(,nrow=K,ncol=1))
#Vectors of mean for each group.
mukilist<-list(matrix(,nrow=K,ncol=1))
ini<-1
end<-ngr[1]
for (i in 1:K) {yki[k]<list(matrix(Dataprove[ini:end,3],nrow=ngr[i],ncol=1))
mukilist[k]<list(matrix(muki[ini:end],nrow=ngr[i],ncol=1))
ini<-end+1
end<-ngr[i+1]+end}

#Vector difference response and mean.
ykiminusmuki<-list(matrix(,nrow=K,ncol=1))
for (i in 1:K) {ykiminusmuki[k]<-(yki[k]-mukilist[k])}

#Matrix Dk for each group.
dkilist<-list(matrix(,nrow=K,ncol=1))
for (j in 1:K) 
  for (n in 1:(p+1)) {dkilist[j][n]<-Dki[ini:end,n]} 
  ini<-end+1
  end<-ngr[j+1]+end

#Gradient Individual part.
elementkGr<-list(matrix(,nrow=K,ncol=1))
grlist<-list(matrix(0,nrow=(p+1),ncol=1))
for (i in 1:K) {elementkGr[i]<-t(dkilist[i,]*vki[i]) %*% ykiminusmuki[i]
grlist[i]<-grlist[i]+elementkGr[i]}

#Hessian individual part.
elementkHs<-list(matrix(,nrow=K,ncol=1))
hslist<-list(matrix(0,nrow=(p+1),ncol=(p+1)))
for (i in 1:K) {elementkHs[i]<-t(dkilist[i,]) %*% vki[i] %*% dkilist[i]
hslist[i]<-hslist[i]+elementkHs[i]}

#Outcome for the aggregated data model with combined analytical and aggregated models.
ybar<-matrix(,nrow=1,ncol=K)
dk<-matrix(,nrow=K,ncol=p+1)
ini<-1
end<-ngr[1]
for (i in 1:K) {ybar[1,i]<-(Dataprove[ini,4]-sum(yki[ini:end]))/(Dataprove[ini,5]-ngr[i])
for (j in 1:(p+1)) {dk[i,j]<-sum(dki[ini:end,j])/ngr[i]}}
# Sigma square aggregated part.

\[
\text{sigma2pbk} = \begin{cases} 
\text{matrix}(nrow=K,ncol=1) \\
\text{ini}<-1 \\
\text{end}<-\text{ngr}[1] \\
\text{for } (i \text{ in } 1:K) {\text{sigma2pbk}[i,]<-\max(((\text{Ybar}[i]-\text{muk}[i])^2-(\text{muk}[i]-\text{phik}[i,]/(\text{Dataprove}[\text{ini},5]-\text{ngr}[i])))/(\text{muk}[i]^2-\text{phik}[i,](1/(\text{Dataprove}[\text{ini},5]-\text{ngr}[i]))),0)} \\
\text{ini}<-\text{end}+1 \\
\text{end}<-\text{ngr}[i+1]+\text{end}
\end{cases}
\]

\[
\text{sigma2pb} = \sum(\text{sigma2pbk}[1:K])/K
\]

# Variance aggregated part.

\[
\text{Dkt} = t(\text{Dk}) \\
\text{Vkbar} = \text{matrix}(nrow=1,ncol=K) \\
\text{ElementKarGr} = \text{list(matrix}(nrow=K,ncol=1)) \\
\text{Grarlist} = \text{list(matrix}(0,nrow=(p+1),ncol=1)) \\
\text{ini}<-1 \\
\text{end}<-\text{ngr}[1] \\
\text{for } (i \text{ in } 1:K) {\text{Vkbar}[1,i]<-\text{sigma2pb}*((\text{muk}[i]^2)-(\text{phik}[i,]/(\text{Dataprove}[\text{ini},5]-\text{ngr}[i])))+(\text{muk}[i]-\text{phik}[i,])*(1/(\text{Dataprove}[\text{ini},5]-\text{ngr}[i]))} \\
\text{ElementKarGr}[[i]] = \text{Dkt[,i]}*(1/\text{Vkbar}[i])*\text{Ybar}[i]-\text{muk}[i] \\
\text{Grarlist}[[1]] = \text{Grarlist}[[1]]+\text{ElementKarGr}[[i]] \\
\text{ini}<-\text{end}+1 \\
\text{end}<-\text{ngr}[i+1]+\text{end}
\]

# Gradient aggregated part.

\[
\text{Grar} = \text{matrix}(,nrow=(p+1),ncol=1) \\
\text{for } (i \text{ in } 1:(p+1)) {\text{Grar}[i,1]<-\text{Grarlist}[[1]][i]}
\]

# Hessian aggregated part.

\[
\text{Hsar} = \text{matrix}(,nrow=(p+1),ncol=(p+1)) \\
\text{for } (i \text{ in } 1:(p+1)) {\text{for } (j \text{ in } 1:(p+1)) {\text{Hsar}[i,j]<-\text{Hsarlist}[[1]][i,j]}}
\]

# Gradient, Hessian for the individual and aggregated part combination.

# We return the Gradient (Grpb), Hessian (Hspb), sigma^2 individual.

# part (sigma2re) and sigma^2 aggregated part (sigma2pb).

\[
\text{list(Gradientpb} = \text{Grpb, Hessianpb} = \text{Hspb, sigma2re} = \text{sigma2re, sigma2pb} = \text{sigma2pb})
\]

## Parameter estimates using the Newton-Raphson algorithm for the IRM.

# We suppose the next structure in the dataset: Id (Identification of individual),
# Group (Group number), Y (Individual outcome: 1 death, 0 alive), O (Population's.
# observed deaths), n (Population at risk), Z1 (covariate Z1), Z2 (covariate Z2),
# ..., Zp (covariate Zp).

\[
\text{farem2} = \text{function}(\text{data, tol, maxiter=100, betainitial)}\{
\text{betanew}<-\text{betainitial} \\
\text{betaold}<-\text{betanew}+1 \\
\text{itercount} = 0 \\
\text{# Solutions for the IRM.} \\
\text{while(max(abs((\text{betaold-betanew})/\text{betaold}))>tol)\{ 
\text{q}<-\text{farem}(\text{betanew, data}) \\
\text{betaold}<-\text{betanew}
\}}
\]

186
betanew<-betanew-ginv(q$Hessian)%*%q$Gradient
itercount=itercount+1
if(is.na(q$sigma2re)) itercount=100
if(itercount>maxiter) break
if(q$sigma2re>50) itercount=100
if(itercount>maxiter) break
}
list(betanew=betanew,sigma2re=q$sigma2re, iterN=itercount)
}
fagrem2<-function(data,tol,maxiter=100,betainitial){
betanew<-betainitial
betaold<-betanew+1
itercount=0
#Solutions for the ARM.
while(max(abs((betaold-betanew)/betaold))>tol){
d<-fagrem(betanew,data)
betanew<-betanew-ginv(d$Hessiana)%*%d$Gradienta
itercount=itercount+1
if(is.na(d$sigma2am)) itercount=100
if(itercount>maxiter) break
if(d$sigma2am>50) itercount=100
if(itercount>maxiter) break
}
list(betanew=betanew,sigma2am=d$sigma2am, iterN=itercount)
}
fpbm2<-function(data,tol,maxiter=100,betainitial){
betanewpb<-betainitial
betaold<-betanewpb+1
itercount=0
#Solutions for the PBEE.
while(max(abs((betaold-betanewpb)/betaold))>tol){
e<-fpbm(betanewpb,data)
betanewpb<-betanewpb-ginv(e$Hessianpb)%*%e$Gradientpb
itercount=itercount+1
if(max(e$sigma2re,e$sigma2pb)>50) itercount=100
if(itercount>maxiter) break
if(max(e$sigma2re,e$sigma2pb)>50) itercount=100
if(itercount>maxiter) break
}
list(betanewpb=betanewpb,sigma2re=e$sigma2re,sigma2pb=e$sigma2pb, iterN=itercount)
}
fgenerate2<-function(group,populationsize,samplesize,variance){
#group=number of groups, populationsize= population size.
#samplesize= sample size, variance=within group variance.
K<-group
nk<-populationsize
mk<-samplesize
varwithin<-variance
#Covariate X1ki with ratio (within variance)/(between variance) equal varwithin/1.
Z1kg<-rnorm(K,0,sqrt(1))
X1ki<-t(matrix(rnorm(nk*K,Z1kg,sqrt(varwithin)),nrow=K,ncol=nk))
#Covariate X2ki with ratio (within variance)/(between variance) equal 1/1.
Z2kg<-matrix(rnorm(K,0,sqrt(1)),nrow=K,ncol=1)
X2ki<-t(matrix(rnorm(nk*K,Z2kg,sqrt(1)),nrow=K,ncol=nk))
#Country specific frailties were generated as independent.
#realized  values from a gamma distribution with mean 1.
#and variance sigma^2. The mean of a gamma is shape*scale.
#and the variance is shape^2.
meanhk<-1
varhk<-0.05
shape<-(meanhk^2)/(varhk)
scale<-(varhk)/(meanhk)
hk<-rgamma(K,shape=shape,scale=scale)

#The disease events, yki, were generated by determining.
#whether a uniform random variable was less than.
#hk*exp(gamma0+beta1*X1ki+beta2*X2ki).

gamma0<-3
beta1<-0.2
beta2<-0.2

yki<-matrix(,nrow=nk,ncol=K)
unif<-matrix(runif(nk*K,0,1),nrow=nk,ncol=K)
yki=ifelse(unif<hk*exp(gamma0+beta1*X1ki+beta2*X2ki),1,0)

#selection of random sample of size mk#.
#and organize data to apply functions #.
#farem, fagrem and fpbm #.

datalist<-list(matrix(,nrow=K,ncol=1))
sampledatalist<-list(matrix(,nrow=K,ncol=1))
inj<-1
end<-mk
data<-matrix(,nrow=mk*K,ncol=5)
for (i in 1:K){
datalist[[i]]<-cbind(matrix(c(1:nk),nrow=nk,ncol=1),matrix(yki[,i],nrow=nk,ncol=1),matrix(X1ki[,i],nrow=nk,ncol=1),matrix(X2ki[,i],nrow=nk,ncol=1),matrix(c(i),nrow=nk,ncol=1))
sampledatalist[[i]]<-datalist[[i]][as.matrix(sample(datalist[[i]][1],mk)),]
data[inj:end,]<-sampledatalist[[i]]
inj<-end+1
end<-mk*(i+1)
}

O<-matrix(apply(yki,2,sum),nrow=K,ncol=1)
inj<-1
end<-mk
datatop<-matrix(,nrow=mk*K,ncol=1)
for (i in 1:K) {datatop[inj:end,1]<-O[i,]
inj<-end+1
end<-mk*(i+1)
}
datatop<-cbind(data,datatop,c(nk))
datapop<-c

fsimulationA<-function(seed,Niter,sigma2,K,N){
set.seed(seed)
count1=0
result1=matrix(0,nrow=Niter,ncol=16)
while(count1<Niter){
tempdata1=fgenerate2(K,2000,N,sigma2)
tol<0.001
gg<-glm(YIND~X1ki+X2ki,data=tempdata1,family=binomial)
betaini<-as.vector(c(matrix(gg$coefficients[1]),matrix(gg$coefficients[2]),matrix(gg$coefficients[3]))
temprem2<-farem2(tempdata1,tol,maxiter=50,betaini)
temprem2<-fagrem2(tempdata1,tol,maxiter=50,betaini)
temprem2<-fpbm2(tempdata1,tol,maxiter=50,betaini)
count1=count1+1
result1[count1,1:5]=matrix(c(t(tempdata1$betanew),tempdata1$sigma2re,tempdata1$iterN),nrow=1,ncol=5)
result1[count1,6:10]=matrix(c(t(tempdata1$betanew),tempdata1$sigma2am,tempdata1$sigma2pb,tempdata1$iteration),nrow=1,ncol=6)
print(count1)
}
}
list(result=result1, sigma2=sigma2, K=K, N=N)
}

# 100 groups-100 sample size in each group.
finalresultA100100A.25 = fsimulationA(123, 1000, .25, 100, 100)
save(list = c("finalresultA100100A.25", ".Random.seed"), file = "100100A025.RData")
savedseed = .Random.seed

finalresultA100100A.5 = fsimulationA(savedseed, 1000, .5, 100, 100)
save(list = c("finalresultA100100A.5", ".Random.seed"), file = "100100A05.RData")
savedseed = .Random.seed

finalresultA100100A1 = fsimulationA(savedseed, 1000, 1, 100, 100)
save(list = c("finalresultA100100A1", ".Random.seed"), file = "100100A1.RData")
savedseed = .Random.seed

finalresultA100100A2 = fsimulationA(savedseed, 1000, 2, 100, 100)
save(list = c("finalresultA100100A2", ".Random.seed"), file = "100100A2.RData")
savedseed = .Random.seed

finalresultA100100A4 = fsimulationA(savedseed, 1000, 4, 100, 100)
save(list = c("finalresultA100100A4", ".Random.seed"), file = "100100A4.RData")
savedseed = .Random.seed

finalresultA100100A8 = fsimulationA(savedseed, 1000, 8, 100, 100)
save(list = c("finalresultA100100A8", ".Random.seed"), file = "100100A8.RData")
savedseed = .Random.seed

finalresultA100100A16 = fsimulationA(savedseed, 1000, 16, 100, 100)
save(list = c("finalresultA100100A16", ".Random.seed"), file = "100100A16.RData")

# 100 groups-50 sample size in each group.
finalresultA10050A.25 = fsimulationA(123, 1000, .25, 100, 50)
save(list = c("finalresultA10050A.25", ".Random.seed"), file = "10050A025.RData")
savedseed = .Random.seed

finalresultA10050A.5 = fsimulationA(savedseed, 1000, .5, 100, 50)
save(list = c("finalresultA10050A.5", ".Random.seed"), file = "10050A05.RData")
savedseed = .Random.seed

finalresultA10050A1 = fsimulationA(savedseed, 1000, 1, 100, 50)
save(list = c("finalresultA10050A1", ".Random.seed"), file = "10050A1.RData")
savedseed = .Random.seed

finalresultA10050A2 = fsimulationA(savedseed, 1000, 2, 100, 50)
save(list = c("finalresultA10050A2", ".Random.seed"), file = "10050A2.RData")
savedseed = .Random.seed

finalresultA10050A4 = fsimulationA(savedseed, 1000, 4, 100, 50)
save(list = c("finalresultA10050A4", ".Random.seed"), file = "10050A4.RData")
savedseed = .Random.seed

finalresultA10050A8 = fsimulationA(savedseed, 1000, 8, 100, 50)
save(list = c("finalresultA10050A8", ".Random.seed"), file = "10050A8.RData")
savedseed = .Random.seed

finalresultA10050A16 = fsimulationA(savedseed, 1000, 16, 100, 50)
save(list = c("finalresultA10050A16", ".Random.seed"), file = "10050A16.RData")

# 50 groups-100 sample size in each group.
finalresultA50100A.25 = fsimulationA(123, 1000, .25, 50, 100)
save(list = c("finalresultA50100A.25", ".Random.seed"), file = "50100A025.RData")
savedseed = .Random.seed

finalresultA50100A.5 = fsimulationA(savedseed, 1000, .5, 50, 100)
save(list = c("finalresultA50100A.5", ".Random.seed"), file = "50100A05.RData")
savedseed = .Random.seed

finalresultA50100A1 = fsimulationA(savedseed, 1000, 1, 50, 100)
save(list = c("finalresultA50100A1", ".Random.seed"), file = "50100A1.RData")
savedseed = .Random.seed

finalresultA50100A2 = fsimulationA(savedseed, 1000, 2, 50, 100)
save(list = c("finalresultA50100A2", ".Random.seed"), file = "50100A2.RData")
savedseed = .Random.seed

finalresultA50100A4 = fsimulationA(savedseed, 1000, 4, 50, 100)
save(list = c("finalresultA50100A4", ".Random.seed"), file = "50100A4.RData")
savedseed = .Random.seed
```r
finalresultA50100A8 <- fSimulationA(savedseed, 1000, 8, 50, 100)
save(list = c("finalresultA50100A8", ".Random.seed"), file = "50100A8.RData")
savedseed = .Random.seed
finalresultA50100A16 <- fSimulationA(savedseed, 1000, 16, 50, 100)
save(list = c("finalresultA50100A16", ".Random.seed"), file = "50100A16.RData")

#50 groups-50 sample size in each group.
finalresultA5050A.25 <- fSimulationA(123, 1000, .25, 50, 50)
save(list = c("finalresultA5050A.25", ".Random.seed"), file = "5050A025.RData")
savedseed = .Random.seed
finalresultA5050A.5 <- fSimulationA(savedseed, 1000, .5, 50, 50)
save(list = c("finalresultA5050A.5", ".Random.seed"), file = "5050A05.RData")
savedseed = .Random.seed
finalresultA5050A1 <- fSimulationA(savedseed, 1000, 1, 50, 50)
save(list = c("finalresultA5050A1", ".Random.seed"), file = "5050A1.RData")
savedseed = .Random.seed
finalresultA5050A2 <- fSimulationA(savedseed, 1000, 2, 50, 50)
save(list = c("finalresultA5050A2", ".Random.seed"), file = "5050A2.RData")
savedseed = .Random.seed
finalresultA5050A4 <- fSimulationA(savedseed, 1000, 4, 50, 50)
save(list = c("finalresultA5050A4", ".Random.seed"), file = "5050A4.RData")
savedseed = .Random.seed
finalresultA5050A8 <- fSimulationA(savedseed, 1000, 8, 50, 50)
save(list = c("finalresultA5050A8", ".Random.seed"), file = "5050A8.RData")
savedseed = .Random.seed
finalresultA5050A16 <- fSimulationA(savedseed, 1000, 16, 50, 50)
save(list = c("finalresultA5050A16", ".Random.seed"), file = "5050A16.RData")

#100 groups-100 sample size in each group.
write.table(finalresultA100100A.25$result, "dataA1var1.txt", row.names = F)  # variance 0.25.
write.table(finalresultA100100A.5$result, "dataA1var2.txt", row.names = F)  # variance 0.5.
write.table(finalresultA100100A.1$result, "dataA1var3.txt", row.names = F)   # variance 1.
write.table(finalresultA100100A.4$result, "dataA1var5.txt", row.names = F)   # variance 4.
write.table(finalresultA100100A16$result, "dataA1var7.txt", row.names = F)  # variance 16.

#50 groups-100 sample size in each group.
write.table(finalresultA50100A.25$result, "dataA2var1.txt", row.names = F)  # variance 0.25.
write.table(finalresultA50100A.5$result, "dataA2var2.txt", row.names = F)  # variance 0.5.
write.table(finalresultA50100A.1$result, "dataA2var3.txt", row.names = F)   # variance 1.
write.table(finalresultA50100A.2$result, "dataA2var4.txt", row.names = F)   # variance 2.
write.table(finalresultA50100A.4$result, "dataA2var5.txt", row.names = F)   # variance 4.
write.table(finalresultA50100A8$result, "dataA2var6.txt", row.names = F)    # variance 8.
write.table(finalresultA50100A16$result, "dataA2var7.txt", row.names = F)  # variance 16.

#100 groups-50 sample size in each group.
write.table(finalresultA10050A.25$result, "dataA3var1.txt", row.names = F)  # variance 0.25.
write.table(finalresultA10050A.5$result, "dataA3var2.txt", row.names = F)  # variance 0.5.
write.table(finalresultA10050A1$result, "dataA3var3.txt", row.names = F)   # variance 1.
write.table(finalresultA10050A2$result, "dataA3var4.txt", row.names = F)   # variance 2.
write.table(finalresultA10050A4$result, "dataA3var5.txt", row.names = F)   # variance 4.
write.table(finalresultA10050A8$result, "dataA3var6.txt", row.names = F)    # variance 8.
write.table(finalresultA10050A16$result, "dataA3var7.txt", row.names = F)  # variance 16.

#100 groups-100 sample size in each group.
write.table(finalresultA10050A.25$result, "dataA4var1.txt", row.names = F)  # variance 0.25.
write.table(finalresultA10050A.5$result, "dataA4var2.txt", row.names = F)  # variance 0.5.
write.table(finalresultA10050A1$result, "dataA4var3.txt", row.names = F)   # variance 1.
write.table(finalresultA10050A2$result, "dataA4var4.txt", row.names = F)   # variance 2.
write.table(finalresultA10050A4$result, "dataA4var5.txt", row.names = F)   # variance 4.
write.table(finalresultA10050A8$result, "dataA4var6.txt", row.names = F)    # variance 8.
write.table(finalresultA10050A16$result, "dataA4var7.txt", row.names = F)  # variance 16.
```
A.3.2 SCC simulation program.

```r
library(MASS)

####################################################.
##FUNCTION INDIVIDUAL RANDOM EFFECTS MODEL (farem).##
##FOR COMPUTE THE GRADIENT, HESSIAN AND SIGMA^2.##
####################################################.

farem<-function(betanew,data) {
  #we suposse the next structure in the dataset:Id (Identification of individual),
  #Group (Group number), Y (Individual outcome: 1 death, 0 alive), O (Population's.
  #observed deaths), n (Population at risk), Z1 (covariate Z1), Z2 (covariate Z2),
  #...,Zp (covariate Zp).
  Dataprove=data
  K<-dim(Dataprove)[1]
  p<-dim(Dataprove)[2]-5
  ngr<-matrix(,nrow=1,ncol=K)
  for (i in 1:K) ngr[1,i]<-dim(subset(Dataprove,Dataprove[,2]==i))[1]
  gamma0<-betanew[1]
  beta=betanew[-1]
  Yki<-matrix(,nrow=N,ncol=1)
  Yki[,1]<-Dataprove[,3]
  muki<-matrix(,nrow=N,ncol=1)
  muki[,1]<-exp(gamma0+as.vector(as.matrix(Dataprove[,6:(5+p)])%*%beta))
  Dki<-matrix(,nrow=N,ncol=p+1)
  for (j in 1:p){
    Dki[,j+1]<-as.numeric(Dataprove[,5+j])*muki[,1]
  }
  muki2<-matrix(,nrow=N,ncol=1)
  muki2[,1]<-muki[,1]^2
  Yaver<-matrix(,nrow=1,ncol=K)
  muk<-matrix(,nrow=1,ncol=K)
  phik<-matrix(,nrow=K,ncol=1)
  sigma2rek<-matrix(,nrow=K,ncol=1)
  ini<-1
  end<-ngr[1]
  for (i in 1:K) {
    Yaver[i,1]<-sum(Yki[ini:end])/ngr[i]
    muk[i,1]<-sum(muki[ini:end])/ngr[i]
    phik[i,1]<-sum(muki2[ini:end])/ngr[i]
    sigma2rek[i,1]<-max((Yaver[i,1]*(Yaver[i,1]*ngr[i]-2*ngr[i]*muk[i,1]-1)+2*((t(muki[ini:end,1])%*%Yki[ini:end,1])/ngr[i])/(ngr[i]*muki[,1]^2)-phik[i,1]+1,-100)
  }
  sigma2re<-sum(sigma2rek[1:K])/K
  i<-1
  end<-ngr[1]
  for (i in 1:K) {
    muk[i,1]<-sum(muki[ini:end])/ngr[i]
    phik[i,1]<-sum(muki2[ini:end])/ngr[i]
    sigma2rek[i,1]<-max((Yaver[i,1]*(Yaver[i,1]*ngr[i]-2*ngr[i]*muk[i,1]-1)+2*((t(muki[ini:end,1])%*%Yki[ini:end,1])/ngr[i])/(ngr[i]*muki[,1]^2)-phik[i,1]+1,-100)
  }
  i<-1
  end<-ngr[1]
  for (i in 1:K) {
    sumk[i,1]<-sum(muki[ini:end]^2)/muki[ini:end]*(1-(1+sigma2re)*muki[ini:end]))
  }
  #Finally, we define the elements of the inverse of V.
  Vki<-list(matrix(nrow=N,ncol=1))
}
for (j in 1:K)
  Vki[[j]]<-matrix(0,nrow=ngr[1,j],ncol=ngr[1,j])
for (i in 1:ngr[1,j])
  yy=1-(1+sigma2re)*(muki[i])
  Vki[[j]][i,(i:ngr[1,j])]<-(sigma2re*(1/yy)*1/(1+sigma2re*muki[[i]]))*(1/(1+sigma2re*sumk[1,j]))
  Vki[[j]][i,i]<-(1/(muki[i]*yy))-(sigma2re*(1/yy)^2)*(1/(1+sigma2re*sumk[1,j]))
Vki[[j]]=Vki[[j]]+t(Vki[[j]])-diag(diag(Vki[[j]]))

###########################.
#Gradient, Hessian for IRM#
###########################.

#Vectors of individual responses for each group. For ngr[4] is NA but we don't use it.
Ykilist<-list(matrix(,nrow=K,ncol=1))

#Vectors of mean for each group.
mukilist<-list(matrix(,nrow=K,ncol=1))
ini<-1
end<-ngr[1]
for (i in 1:K) {Ykilist[[i]]<-list(matrix(Dataprove[ini:end,3],nrow=ngr[i],ncol=1))
mukilist[[i]]<-list(matrix(muki[ini:end],nrow=ngr[i],ncol=1))
  ini<-end+1
  end<-ngr[i+1]+end}

#Vector difference response and mean.
Ykiminusmuki<-list(matrix(,nrow=K,ncol=1))
for (i in 1:K) {Ykiminusmuki[[i]]<-(Ykilist[[i]]-mukilist[[i]])}

#Matrix Dk for each group.
Dkilist<-list(matrix(,nrow=K,ncol=1))
for (i in 1:K) Dkilist[[i]]<-matrix(,nrow=ngr[1,i],ncol=p+1)
ini<-1
end<-ngr[1]
for (j in 1:K){
  for (n in 1:(p+1)){
    Dkilist[[j]][,n]<-Dki[ini:end,n]
    ini<-end+1
    end<-ngr[j+1]+end}
}

#Gradient.
ElementKGr<-list(matrix(,nrow=K,ncol=1))
Grlist<-list(matrix(0,nrow=(p+1),ncol=1))
for (i in 1:K) {ElementKGr[[i]]<-t(Dkilist[[i]])%*%Vki[[i]]%*%Ykiminusmuki[[i]]
  Grlist[[1]]<-Grlist[[1]]+ElementKGr[[i]]}
Gr<-matrix(,nrow=(p+1),ncol=1)
for (i in 1:(p+1)) {Gr[i,1]<-Grlist[[1]]

#Hessian.
ElementKhs<-list(matrix(,nrow=K,ncol=1))
Hslist<-list(matrix(0,nrow=(p+1),ncol=(p+1)))
for (i in 1:K) {ElementKhs[[i]]<-1*t(Dkilist[[i]])%*%Vki[[i]]%*%Dkilist[[i]]
  Hslist[[1]]<-Hslist[[1]]+ElementKhs[[i]]}
Hs<-matrix(,nrow=(p+1),ncol=(p+1))
for (j in 1:(p+1)){
  for (i in 1:(p+1)) {Hs[i,j]<-Hslist[[1]][i,j]}}

#We return the Gradient (Gr), Hessian(Hs) and sigma^2(sigma2re).
list(Gradient=Gr,Hessian=Hs,sigma2re=sigma2re)}
## Data prove = data

#K is the number of groups. We suppose that groups are ordered and they have all. The correlatives numbers. For example: 1, 2, 3 and not 1, 3 (There are no number 2).

N <- dim(Dataprove)[1]
K <- Dataprove[N, 2]
p <- dim(Dataprove)[2] - 5

ngr <- matrix(, nrow = 1, ncol = K)
for (i in 1:K) ngr[i, i] <- dim(subset(Dataprove, Dataprove[2] == i))[1]

# Indivdual mean (muki).
muki <- matrix(, nrow = N, ncol = 1)
muki[, 1] <- exp(gamma0 + as.vector(as.matrix(Dataprove[, 6:(5 + p)]) * %*% beta))

# Individual matrix D.
Dki <- matrix(, nrow = N, ncol = p + 1)
Dki[, 1] <- muki[, 1]
for (j in 1:p) {
  Dki[, j + 1] <- as.numeric(Dataprove[, 5 + j]) * muki[, 1]
}

# Variance for the ARM.
muki2 <- matrix(, nrow = N, ncol = 1)
muki2[, 1] <- muki[, 1]^2

# Outcome for the ARM as defined in Sheppard and Prentice (Biometrics, 1995).
Y <- matrix(, nrow = 1, ncol = K)
muk <- matrix(, nrow = 1, ncol = K)
phik <- matrix(, nrow = K, ncol = 1)

# Matrix D for the ARM.
Dk <- matrix(, nrow = K, ncol = p + 1)

# First, we compute sigma square.
sigma2amk <- matrix(, nrow = K, ncol = 1)
ini <- 1
end <- ngr[1]
for (i in 1:K) {
  Y[i, 1] <- ((Dataprove[ini, 4]) / (Dataprove[ini, 5]))
  muk[i, 1] <- sum(muki[ini:end]) / ngr[i]
  phik[i, 1] <- sum(muki2[ini:end]) / ngr[i]
  for (j in 1:(p + 1)) {
    Dki[i, j] <- sum(Dki[ini:end, j]) / ngr[i]
    sigma2amk[i, 1] <- min((Y[i, 1] - muk[i, 1])^2 - phik[i, 1] * muk[i, 1]^2), -100)
    ini <- ini + 1
  }
  end <- ngr[i + 1] + end
}
sigma2am <- sum(sigma2amk[1:K]) / K

# Finally, we define the variance.
Vk <- matrix(, nrow = 1, ncol = K)
in1 <- 1
end <- ngr[1]
for (i in 1:K) {
  Vk[i, 1] <- sigma2am * (muk[i, 1]^2) - phik[i, 1] / (Dataprove[ini, 5])) + (muk[i] - phik[i, 1]) / (Dataprove[ini, 5])
  ini <- ini + 1
  end <- ngr[i + 1]
}

# Gradient, Hessian for ARM.
Dk <- t(Dk)
ElementtKdGr <- list(matrix(, nrow = K, ncol = 1))
Gralist <- list(matrix(0, nrow = (p + 1), ncol = 1))
for (i in 1:K) {
  ElementtKdGr[i] <- Dki[i, 1] * (1 / Vk[i]) * (muk[i] - muk[i, 1])
}

Gra <- matrix(, nrow = (p + 1), ncol = 1)
for (i in 1:(p + 1)) {
  Gra[i, 1] <- Gralist[[1]][i]
}

# Hessian.
ElementtKhs <- list(matrix(, nrow = K, ncol = 1))
Hsalist <- list(matrix(0, nrow = (p + 1), ncol = 1))
for (i in 1:K) {
  ElementtKhs[i] <- 1 * (muk[i] - muk[i, 1]) * (1 / Vk[i]) * ElementtKdGr[i]
}

Hsa <- matrix(, nrow = (p + 1), ncol = (p + 1))
for (i in 1:(p + 1)) {
  for (j in 1:(p + 1)) {
    Hsa[i, j] <- Hsalist[[1]][i, j]
  }
}
# We return the Gradient (Gra), Hessian (Hsa) and sigma^2 (sigma2am).
list(Gradienta=Gra, Hessiana=Hsa, sigma2am=sigma2am)}

function (betanew, data) {
  # We suppose the next structure in the dataset: Id (Identification of individual).
  # Group (Group number), Y (Individual outcome: 1 death, 0 alive), O (Population's observed deaths), n (Population at risk), Z1 (covariate Z1), Z2 (covariate Z2), ... , Zp (covariate Zp).
  Dataprove = data
  K = dim(Dataprove)[2]
  N = dim(Dataprove)[1]
  p = dim(Dataprove)[2] - 5
  ngr = matrix(, nrow = 1, ncol = K)
  for (i in 1:K) ngr[, i] = dim(subset(Dataprove, Dataprove[2] == i))[1]
  gamma0 = betanew[1]
  beta = betanew[-1]

  # Gradient, Hessian for the individual part.
  # Individual outcome.
  Yki = matrix(, nrow = N, ncol = 1)
  Yki[, 1] = Dataprove[, 3]
  # Individual mean.
  muki = matrix(, nrow = N, ncol = 1)
  muki[, 1] = exp(gamma0 + as.vector(as.matrix(Dataprove[, 6:(5 + p)]) %*% beta))
  # Individual matrix D.
  Dki = matrix(, nrow = N, ncol = p + 1)
  Dki[, 1] = muki[, 1]
  for (j in 1:p) {Dki[, j + 1] = as.numeric(Dataprove[, 5 + j]) * muki[, 1]}

  # Inverse variance-covariance matrix individual part.
  # First, we compute sigma square for the individual part.
  muki2 = matrix(, nrow = N, ncol = 1)
  muki2[, 1] = muki[, 1]^2
  Yaver = matrix(, nrow = 1, ncol = K)
  muk = matrix(, nrow = 1, ncol = K)
  phik = matrix(, nrow = K, ncol = 1)
  sigma2rek = matrix(, nrow = K, ncol = 1)
  ini = 1
  end = ngr[, 1]
  for (i in 1:K) {Yaver[, i] = sum(Yki[, ini:end])/ngr[i]
    muk[, i] = sum(muki[, ini:end])/ngr[i]
    phik[, i] = sum(muki2[, ini:end])/ngr[i]
    sigma2rek[, i] = max((Yaver[, i] * (Yaver[, i] * sigma2rek[, i] - 2 * ngr[i] * muk[, i] - 1) + 2 * (t(muki[, ini:end, i]) %*% Yki[, ini:end])/ngr[i])/(ngr[i] * (muk[, i]^2 - phik[, i])) + 1, -100)
    ini = end + 1
    end = ngr[, i+1] + end}

  sigma2rek = sum(sigma2rek[1:K])/K

  # We compute the expression for one part (transpose(muk) * inverse(Deltak) * muk).
  sumk = matrix(, nrow = 1, ncol = K)
  ini = 1
  end = ngr[, 1]
  for (i in 1:K) {sumk[, i] = sum((muki[, ini:end]^2)/(muki[, ini:end] * (1 - (1 + sigma2rek) * muki[, ini:end])))
    ini = end + 1
    end = ngr[, i+1] + end}

  ...
# Finally we define the elements of the inverse of V.

\[ V_{ki} <- list(matrix(, nrow=K, ncol=1)) \]

for (i in 1:K) {
    Vki[i] <- matrix(0, nrow=ngr[1,i], ncol=ngr[1,i])
    for (j in 1:ngr[1,i]) {
        yy <- 1 - (1 + sigma2re)*muki[i]
        Vki[i][j] <- (-sigma2re*(1/yy)*(1/(1 - (1 + sigma2re)*muki[j])))^2*(1/(1 + sigma2re*sumk[1,j]))
        Vki[i][i] <- (-sigma2re*(1/yy)^2*(1/(1 + sigma2re*sumk[1,j])))
    }
    Vki[i] = Vki[i] + t(Vki[i]) - diag(diag(Vki[i]))
}

# Vectors of individual responses for each group. For ngr[4] is NA but we don't use it.
Ykilist <- list(matrix(, nrow=K, ncol=1))

# Vectors of mean for each group.
mukilist <- list(matrix(, nrow=K, ncol=1))
ini <- 1
end <- ngr[1]
for (i in 1:K) {
    Ykilist[i] <- list(matrix(Dataprove[ini:end,3], nrow=ngr[i], ncol=1))
    mukilist[i] <- list(matrix(muki[ini:end], nrow=ngr[i], ncol=1))
    ini <- end + 1
    end <- ngr[i + 1] + end
}

# Vector difference response and mean.
Ykminusmuki <- list(matrix(, nrow=K, ncol=1))
for (i in 1:K) {
    Ykminusmuki[i] <- (Ykilist[i] - mukilist[i])
}

# Matrix Dk for each group.
Dkilist <- list(matrix(, nrow=K, ncol=1))
ini <- 1
end <- ngr[1]
for (i in 1:K) {
    for (n in 1:(p+1)) {
        Dkilist[i][n] <- Dki[ini:end, n]
    } 
    ini <- end + 1
    end <- ngr[i + 1] + end
}

# Gradient Individual part.
ElementKGr <- list(matrix(, nrow=K, ncol=1))
Grlist <- list(matrix(0, nrow=(p+1), ncol=1))
for (i in 1:K) {
    ElementKGr[i] <- t(Dkilist[i])%*%Vki[i]%*%Ykminusmuki[i]
    Grlist[i] <- Grlist[i] + ElementKGr[i]
}
Gr <- matrix(, nrow=(p+1), ncol=1)
for (i in 1:(p+1)) {
    Gr[i, 1] <- Grlist[i][1]
}

# Hessian individual part.
ElementKHS <- list(matrix(, nrow=K, ncol=1))
Hslist <- list(matrix(0, nrow=(p+1), ncol=(p+1)))
for (i in 1:K) {
    ElementKHS[i] <- 1%*%t(Dkilist[i])%*%Vki[i]%*%Dkilist[i]
    Hslist[i] <- Hslist[i] + ElementKHS[i]
}
Hs <- matrix(, nrow=(p+1), ncol=(p+1))
for (i in 1:(p+1)) {
    for (j in 1:(p+1)) {
        Hs[i, j] <- Hslist[i][i, j]
    }
}

# Outcome for the aggregated data model with combined analytical and aggregated models.
Ybar <- matrix(, nrow=1, ncol=K)

# Matrix D for the aggregated part.
Dk <- matrix(, nrow=K, ncol=p+1)
ini <- 1
end <- ngr[1]
for (i in 1:K) {
    for (j in 1:(p+1)) {
        Dk[i,j] <- sum(Dki[ini:end,j]) / ngr[i]
    }
    ini <- end + 1
    end <- ngr[i + 1] + end
}

# Sigma square aggregated part.
sigma2pbk <- matrix(, nrow = K, ncol = 1)
ini <- 1
end <- ngr[1]
for (i in 1:K) {
  ini <- end + 1
  end <- ngr[i+1] + end}

sigma2pb <- sum(sigma2pbk[1:K])/K

# Variance aggregated part.
Dkt <- t(Dk)
Vktbar <- matrix(, nrow = 1, ncol = K)
ElementKarGr <- list(matrix(, nrow = K, ncol = 1))
Grarlist <- list(matrix(0, nrow = (p+1), ncol = 1))
ini <- 1
end <- ngr[1]
for (i in 1:K) {
  Vktbar[1,i] <- sigma2pb*(muk[i]^2 - (phik[i,]/(Dataprove[ini,5] - ngr[i]))) + (muk[i] - phik[i,])*(1/(Dataprove[ini,5] - ngr[i]))
  ElementKarGr[[i]] <- Dkt[,i] * (1/Vktbar[i]) * (Ybar[i] - muk[i])
  Grarlist[[1]] <- Grarlist[[1]] + ElementKarGr[[i]]
  ini <- end + 1
  end <- ngr[i+1] + end}

# Gradient aggregated part.
Grar <- matrix(, nrow = (p+1), ncol = 1)
for (i in 1:(p+1)) {Grar[i,1] <- Grarlist[[1]][i]}

# Hessian aggregated part.
ElementKarHs <- list(matrix(, nrow = K, ncol = 1))
Hsarlist <- list(matrix(0, nrow = (p+1), ncol = (p+1)))
for (i in 1:K) {ElementKarHs[[i]] <- t(Dkt[,i]) * (1/Vktbar[i]) * Dk[i,]
  Hsarlist[[1]] <- Hsarlist[[1]] + ElementKarHs[[i]]}
Hsar <- matrix(, nrow = (p+1), ncol = (p+1))
for (i in 1:(p+1)) {for (j in 1:(p+1)) {Hsar[i,j] <- Hsarlist[[1]][i,j]}}

# Gradient, Hessian for the individual and aggregated part combination.
# Gradient PBEE.
Grpb <- (Gr + Grar)
# Hessian PBEE.
Hspb <- (Hs + Hsar)
list(Gradientpb = Grpb, Hessianpb = Hspb, sigma2re = sigma2re, sigma2pb = sigma2pb)
if(is.na(q$sigma2re)) itercount=100
if(itercount>maxiter) break
if(q$sigma2re>50) itercount=100
if(itercount>maxiter) break
}
list(betanew=betanew,sigma2re=q$sigma2re, iterN=itercount)
}
fagrem2<-function(data,tol,maxiter=100,betainitial){
betanewa<-betainitial
betaold<-betanewa+1
itercount=0
Hdim=length(betanewa)^2
#Solutions for the ARM.
while(max(abs((betaold-betanewa)/betaold))>tol){
dc<-fagrem(betanewa,data)
if(sum(is.finite(d$Hessiana))<Hdim) itercount=999
if(itercount>maxiter) break
betaold<-betanewa-ginv(d$Hessiana)%*%d$Gradienta
itercount=itercount+1
if(is.na(d$sigma2am)) itercount=100
if(itercount>maxiter) break
if(d$sigma2am>50) itercount=100
if(itercount>maxiter) break
}
list(betanewa=betanewa,sigma2am=d$sigma2am, iterN=itercount)
}
fpbm2<-function(data,tol,maxiter=100,betainitial){
betanewpb<-betainitial
betaold<-betanewpb+1
itercount=0
Hdim=length(betanewpb)^2
#Solutions for the PBEE.
while(max(abs((betaold-betanewpb)/betaold))>tol){
ec<-fpbm(betanewpb,data)
betanewpb<-betanewpb-ginv(e$Hessianpb)%*%e$Gradientpb
itercount=itercount+1
if(is.na(max(e$sigma2re,e$sigma2pb))) itercount=100
if(itercount>maxiter) break
if(max(e$sigma2re,e$sigma2pb)>50) itercount=100
if(itercount>maxiter) break
}
list(betanewpb=betanewpb,sigma2re=e$sigma2re,sigma2pb=e$sigma2pb, iterN=itercount)
}
fgenerate2<-function(group,populationsize,samplesize,variance){
#group=number of groups, populationsize= population size.
#samplesize= sample size, variance=within group variance.
K<group
nk<-populationsize
mk<samplesize
varwithin<variance
#Covariate X1ki & X2ki. They are correlated 0.3 at the community.
and individual levels (see Prentice & Sheppard).
zk=mvrnorm(K,c(0,0),matrix(c(1,.3,.3,1),2,2))
cov=0.3*sqrt(varwithin)*1
covm=matrix(c(varwithin,cov,cov,1),2,2)
X1ki=matrix(0,nk,K)
X2ki=matrix(0,nk,K)
for(i in 1:K){
  znk=mvrnorm(nk,zk[i,],covm)
  X1ki[,i]=znk[,1]
\text{x2ki[,i]=znk[,2]}

#Country specific frailties were generated as independent.
#realized values from a gamma distribution with mean 1.
#and variance \( \sigma^2 \). The mean of a gamma is \( \text{shape} \times \text{scale} \).
#and the variance is \( \text{shape} \times (\text{scale}^2) \).

\text{meanhk<-1}
\text{varhk<-0.05}
\text{shape<-(meanhk^2)/(varhk)}
\text{scale<-(varhk)/(meanhk)}
\text{hk<-rgamma(K,shape=shape,scale=scale)[,]}
\text{hk=t(matrix(rep(hk,nk),nrow=K,ncol=nk))}

#The disease events, \( y_{ki} \), were generated by determining.
#whether a uniform random variable was less than.
#\( h_k \times \exp(\gamma_0+\beta_1 X_{1ki}+\beta_2 X_{2ki}) \).

\text{gamma0<-3}
\text{beta1<-0.2}
\text{beta2<-0.2}
\text{yki<-matrix(,nrow=nk,ncol=K)}
\text{unif<-matrix(runif(nk*K,0,1),nrow=nk,ncol=K)}
\text{yki=ifelse(unif<hk*exp(gamma0+beta1*X1ki+beta2*X2ki),1,0)}

#selection of random sample of size \( m_k \).
#and organize data to apply functions #.
#farem, fagrem and fpbm #.

\text{datalist<-list(matrix(,nrow=K,ncol=1))}
\text{sampledatalist<-list(matrix(,nrow=K,ncol=1))}
\text{ini<-1}
\text{end<-mk}
\text{data<-matrix(,nrow=mk*K,ncol=5)}
\text{for (i in 1:K)}{
  \text{datalist[[i]]<-}
  \text{cbind(matrix(c(1:nk),nrow=nk,ncol=1),matrix(yki[,i],nrow=nk,ncol=1),matrix(X1ki[,i],nrow=nk,ncol=1),matrix(X2ki[,i],nrow=nk,ncol=1),matrix(c(i),nrow=nk,ncol=1))}
  \text{sampledatalist[[i]]<-as.matrix(sample(datalist[[i]][,1],mk),)}
  \text{data[ini:end,]<-sampledatalist[[i]]}
  \text{ini<-end+1}
  \text{end<-mk*(i+1)}
}
\text{O<-matrix(apply(yki,2,sum),nrow=K,ncol=1)}
\text{ini<-1}
\text{end<-mk}
\text{datapop<-matrix(,nrow=mK*K,ncol=1)}
\text{for (i in 1:K) {datapop[ini:end,]<-O[i,]}}
\text{ini<-end+1}
\text{end<-mk*(i+1)}
\text{datafin<-cbind(data,datapop,c(nk))}
\text{datafin<-data.frame(id=matrix(datafin[,1]),group=matrix(datafin[,5]),YIND=matrix(datafin[,2]),O=matrix(datafin[,6]),n=matrix(datafin[,7]),X1ki=matrix(datafin[,3]),X2ki=matrix(datafin[,4]))}

#Simulation results##.
result[count1,1:5]=matrix(c(t(tempdata1a1$betanew),tempdata1a1$sigma2re,tempdata1a1$iterN),nrow=1,ncol=5)
result[count1,6:10]=matrix(c(t(tempdata1a2$betanewa),tempdata1a2$sigma2am,tempdata1a2$iterN),nrow=1,ncol=5)
result[count1,11:16]=matrix(c(t(tempdata1a3$betanewpb),tempdata1a3$sigma2re,tempdata1a3$sigma2pb,tempdata1a3$iterN),nrow=1,ncol=6)
print(count1)
}
list(result=result1,sigma2=sigma2,K=K,N=N)

#100 groups-100 sample size in each group.
finalresultB100100B.25=fsimulationB(123,1000,.25,100,100)
save(list=c("finalresultB100100B.25",".Random.seed"),file="100100B025.RData")
savedseed=.Random.seed
finalresultB100100B.5=fsimulationB(savedseed,1000,.5,100,100)
save(list=c("finalresultB100100B.5",".Random.seed"),file="100100B05.RData")
savedseed=.Random.seed
finalresultB100100B1=fsimulationB(savedseed,1000,1,100,100)
save(list=c("finalresultB100100B1",".Random.seed"),file="100100B1.RData")
savedseed=.Random.seed
finalresultB100100B2=fsimulationB(savedseed,1000,2,100,100)
save(list=c("finalresultB100100B2",".Random.seed"),file="100100B2.RData")
savedseed=.Random.seed
finalresultB100100B4=fsimulationB(savedseed,1000,4,100,100)
save(list=c("finalresultB100100B4",".Random.seed"),file="100100B4.RData")
savedseed=.Random.seed
finalresultB100100B8=fsimulationB(savedseed,1000,8,100,100)
save(list=c("finalresultB100100B8",".Random.seed"),file="100100B8.RData")
savedseed=.Random.seed
finalresultB100100B16=fsimulationB(savedseed,1000,16,100,100)
save(list=c("finalresultB100100B16",".Random.seed"),file="100100B16.RData")

#100 groups-50 sample size in each group.
finalresultB10050B.25=fsimulationB(123,1000,.25,100,50)
save(list=c("finalresultB10050B.25",".Random.seed"),file="10050B025.RData")
savedseed=.Random.seed
finalresultB10050B.5=fsimulationB(savedseed,1000,.5,100,50)
save(list=c("finalresultB10050B.5",".Random.seed"),file="10050B05.RData")
savedseed=.Random.seed
finalresultB10050B1=fsimulationB(savedseed,1000,1,100,50)
save(list=c("finalresultB10050B1",".Random.seed"),file="10050B1.RData")
savedseed=.Random.seed
finalresultB10050B2=fsimulationB(savedseed,1000,2,100,50)
save(list=c("finalresultB10050B2",".Random.seed"),file="10050B2.RData")
savedseed=.Random.seed
finalresultB10050B4=fsimulationB(savedseed,1000,4,100,50)
save(list=c("finalresultB10050B4",".Random.seed"),file="10050B4.RData")
savedseed=.Random.seed
finalresultB10050B8=fsimulationB(savedseed,1000,8,100,50)
save(list=c("finalresultB10050B8",".Random.seed"),file="10050B8.RData")
savedseed=.Random.seed
finalresultB10050B16=fsimulationB(savedseed,1000,16,100,50)
save(list=c("finalresultB10050B16",".Random.seed"),file="10050B16.RData")

#50 groups-100 sample size in each group.
finalresultB50100B.25=fsimulationB(123,1000,.25,50,100)
save(list=c("finalresultB50100B.25",".Random.seed"),file="50100B025.RData")
savedseed=.Random.seed
finalresultB50100B.5=fsimulationB(savedseed,1000,.5,50,100)
save(list=c("finalresultB50100B.5",".Random.seed"),file="50100B05.RData")
savedseed=.Random.seed
finalresultB50100B1=fsimulationB(savedseed,1000,1,50,100)
save(list=c("finalresultB50100B1",".Random.seed"),file="50100B1.RData")
savedseed=.Random.seed

199
finalresultB50100B2=fsimulationB(savedseed,1000,2,50,100)
save(list=c("finalresultB50100B2",".Random.seed"),file="50100B2.RData")
savedseed=.Random.seed

finalresultB50100B4=fsimulationB(savedseed,1000,4,50,100)
save(list=c("finalresultB50100B4",".Random.seed"),file="50100B4.RData")
savedseed=.Random.seed

finalresultB50100B8=fsimulationB(savedseed,1000,8,50,100)
save(list=c("finalresultB50100B8",".Random.seed"),file="50100B8.RData")
savedseed=.Random.seed

finalresultB50100B16=fsimulationB(savedseed,1000,16,50,100)
save(list=c("finalresultB50100B16",".Random.seed"),file="50100B16.RData")

#50 groups-50 sample size in each group.

finalresultB5050B.25=fsimulationB(123,1000,.25,50,50)
save(list=c("finalresultB5050B.25",".Random.seed"),file="5050B025.RData")
savedseed=.Random.seed

finalresultB5050B.5=fsimulationB(savedseed,1000,.5,50,50)
save(list=c("finalresultB5050B.5",".Random.seed"),file="5050B05.RData")
savedseed=.Random.seed

finalresultB5050B1=fsimulationB(savedseed,1000,1,50,50)
save(list=c("finalresultB5050B1",".Random.seed"),file="5050B1.RData")
savedseed=.Random.seed

finalresultB5050B2=fsimulationB(savedseed,1000,2,50,50)
save(list=c("finalresultB5050B2",".Random.seed"),file="5050B2.RData")
savedseed=.Random.seed

finalresultB5050B4=fsimulationB(savedseed,1000,4,50,50)
save(list=c("finalresultB5050B4",".Random.seed"),file="5050B4.RData")
savedseed=.Random.seed

finalresultB5050B8=fsimulationB(savedseed,1000,8,50,50)
save(list=c("finalresultB5050B8",".Random.seed"),file="5050B8.RData")
savedseed=.Random.seed

finalresultB5050B16=fsimulationB(savedseed,1000,16,50,50)
save(list=c("finalresultB5050B16",".Random.seed"),file="5050B16.RData")

#100 groups-100 sample size in each group.

write.table(finalresultB100100B.25$result,"dataB1var1.txt",row.names=F)  #variance 0.25.
write.table(finalresultB100100B.5$result,"dataB1var2.txt",row.names=F)    #variance 0.5.
write.table(finalresultB100100B1$result,"dataB1var3.txt",row.names=F)     #variance 1.
write.table(finalresultB100100B4$result,"dataB1var5.txt",row.names=F)     #variance 4.
write.table(finalresultB100100B8$result,"dataB1var6.txt",row.names=F)     #variance 8.
write.table(finalresultB100100B16$result,"dataB1var7.txt",row.names=F)    #variance 16.

write.table(finalresultB50100B.25$result,"dataB2var1.txt",row.names=F)  #variance 0.25.
write.table(finalresultB50100B.5$result,"dataB2var2.txt",row.names=F)    #variance 0.5.
write.table(finalresultB50100B4$result,"dataB2var5.txt",row.names=F)     #variance 4.
write.table(finalresultB50100B8$result,"dataB2var6.txt",row.names=F)     #variance 8.
write.table(finalresultB50100B16$result,"dataB2var7.txt",row.names=F)    #variance 16.

write.table(finalresultB10050B.25$result,"dataB3var1.txt",row.names=F)  #variance 0.25.
write.table(finalresultB10050B.5$result,"dataB3var2.txt",row.names=F)    #variance 0.5.
write.table(finalresultB10050B1$result,"dataB3var3.txt",row.names=F)     #variance 1.
write.table(finalresultB10050B2$result,"dataB3var4.txt",row.names=F)     #variance 2.
write.table(finalresultB10050B4$result,"dataB3var5.txt",row.names=F)     #variance 4.
write.table(finalresultB10050B8$result,"dataB3var6.txt",row.names=F)     #variance 8.
write.table(finalresultB10050B16$result,"dataB3var7.txt",row.names=F)    #variance 16.

write.table(finalresultB5050B.25$result,"dataB4var1.txt",row.names=F)  #variance 0.25.
write.table(finalresultB5050B.5$result,"dataB4var2.txt",row.names=F)    #variance 0.5.
write.table(finalresultB5050B1$result,"dataB4var3.txt",row.names=F)     #variance 1.
write.table(finalresultB5050B2$result,"dataB4var4.txt",row.names=F)     #variance 2.
write.table(finalresultB5050$result,"dataB4var5.txt",row.names=F) #variance 4.
write.table(finalresultB5050B8$result,"dataB4var6.txt",row.names=F) #variance 8.
write.table(finalresultB5050B16$result,"dataB4var7.txt",row.names=F) #variance 16.

A.3.3 ECC simulation program.

library(MASS)

#FUNCTION INDIVIDUAL RANDOM EFFECTS MODEL (farem)
#FOR COMPUTE THE GRADIENT, HESSIAN AND SIGMA^2

farem<-function(betanew,data) {
  # We suppose the next structure in the dataset: Id (Identification of individual),
  # Group (Group number), Y (Individual outcome: 1 death, 0 alive), O (Population's.
  # Observed deaths), n (Population at risk), Z1 (Covariate Z1), Z2 (Covariate Z2),
  # ..., Zp (Covariate Zp).
  Dataprove=data
  K<-dim(Dataprove)[1]
  N<-dim(Dataprove)[2]
  p<-dim(Dataprove)[2]-5
  for (i in 1:K) ngr[i,1]<-dim(subset(Dataprove,Dataprove[,2]==i))[1]
  gamma0<betanew[1]
  beta=betanew[-1]

  # Individual outcome.
  Yki<-matrix(,nrow=N,ncol=1)
  Yki[,1]<-Dataprove[,3]

  # Individual mean.
  muki<-matrix(,nrow=N,ncol=1)
  muki[,1]<-exp(gamma0+as.vector(as.matrix(Dataprove[,6:(5+p)])%*%beta))

  # Matrix D for the IRM.
  Dki<-matrix(,nrow=N,ncol=p+1)
  Dki[,1]<-muki[,1]
  for (j in 1:p){
    Dki[,j+1]<-as.numeric(Dataprove[,5+j])*muki[,1]
  }

  # Inverse variance-covariance matrix for the IRM.
  muki2<-matrix(,nrow=N,ncol=1)
  muki2[,1]<-muki[,1]^2
  Yaver<-matrix(,nrow=1,ncol=K)
  muk<-matrix(,nrow=1,ncol=K)
  phik<-matrix(,nrow=K,ncol=1)
  sigma2rek<-matrix(,nrow=K,ncol=1)
  ini<-1
  end<-ngr[1]
  for (i in 1:K) {
    Yaver[1,i]<-sum(Yki[ini:end])/ngr[i]
    muk[1,i]<-sum(muki[ini:end])/ngr[i]
    phik[i,1]<-sum(muki2[ini:end])/ngr[i]
    sigma2rek[i,1]<max((Yaver[1,i]/(Yaver[1,i]*ngr[i]+2)*muk[1,i]-1)+2*(t(muki[ini:end,1])%*%Yki[ini:end,1])/ngr[i])/(ngr[i]*muki[1,i]^2)-phik[1,1]+1,-100)
    ini<end+1
    end<-ngr[i+1]
  }
  sigma2re<-sum(sigma2rek[1:K])/K

  # We compute the expression for one part (transpose(muk)*Inverse(Deltak)*muk).
  sumk<-matrix(,nrow=1,ncol=K)
  ini<-1
  end<-ngr[1]
  for (i in 1:K) {
    sumk[i,1]<-sum(((muki[ini:end,1])^2)/(muki[ini:end,1]*(1+sigma2re)*muki[ini:end]))
    ini<end+1
  }
}
end<-ngr[i+1]+end}

#Finally, we define the elements of the inverse of V.
Vki<-list(matrix(,nrow=K,ncol=1))

for (i in 1:K) {
  Vki[[i]]<-matrix(0,nrow=ngr[1,j],ncol=1)
}

for (i in 1:ng[i,j]) {
  y=1-(1+sigma2re)*muki[j]
  Vki[[j]][i,(i:ng[i,j])]<-(sigma2re*(1/yy)*(1/(1+(1+sigma2re)*muki[i,y]))**(1/(1+(1+sigma2re)*sumk[1,j])))
  Vki[[j]][i,i]<-(1/(muki[i,y])**(1+(1+sigma2re)*sumk[1,j]))
  Vki[[j]][i]<-Vki[[j]]+t(Vki[[j]])-diag(diag(Vki[[j]]))
}

###########################.
#Gradient, Hessian for IRM#
###########################.

#Vectors of individual responses for each group. For ngr[4] is NA but we don't use it.
Ykilist<-list(matrix(,nrow=K,ncol=1))

#Vectors of mean for each group.
mukilist<-list(matrix(,nrow=K,ncol=1))

ini<-1
end<-ngr[1]

for (i in 1:K) {Ykilist[i]<-list(matrix(Dataprove[ini:end,3],nrow=ngr[i],ncol=1))
mukilist[i]<-list(matrix(muki[ini:end],nrow=ngr[i],ncol=1))

ini<-end+1
end<-ngr[i+1]+end}

#Vector difference response and mean.
Ykiminusmuki<-list(matrix(,nrow=K,ncol=1))

for (i in 1:K) Ykiminusmuki[i]<-(Ykilist[i]-mukilist[i])

#Matrix Dk for each group.
Dkilist<-list(matrix(,nrow=K,ncol=1))

for (i in 1:K) Dkilist[i]<-matrix(,nrow=ngr[1,i],ncol=p+1)

ini<-1
end<-ngr[1]

for (j in 1:K) {
  for (n in 1:(p+1)) {Dkilist[i][j,n]<-Dki[ini:end,n]
  ini<-end+1
  end<-ngr[j+1]+end}
}

#Gradient.
ElementKGr<-list(matrix(,nrow=K,ncol=1))
Grlist<-list(matrix(0,nrow=(p+1),ncol=1))

for (i in 1:K) {
  ElementKGr[[i]]<-t(Dkilist[[i]])%*%Vki[[i]]%*%Ykiminusmuki[i]
}

Gr<-matrix(,nrow=(p+1),ncol=1)

for (i in 1:(p+1)) {Gr[1,1]<-Grlist[1][i]}

#Hessian.
ElementKHs<-list(matrix(,nrow=K,ncol=1))
Hslist<-list(matrix(0,nrow=(p+1),ncol=(p+1)))

for (i in 1:K) {
  ElementKHs[[i]]<-1+t(Dkilist[[i]])%*%Vki[[i]]%*%Dkilist[[i]]
}

Hs<-matrix(,nrow=(p+1),ncol=(p+1))

for (i in 1:(p+1)) {
  for (j in 1:(p+1)) {Hs[i,j]<-Hslist[1][i,j]}
}

#We return the Gradient (Gr), Hessian(Hs) and sigma^2(sigma2re).
list(Gradient=Gr,Hessian=Hs,sigma2re=sigma2re)}

#####################################################.
##FUNCTION AGGREGATED RANDOM EFFECTS MODEL (fagrem)##.
#####################################################.

fagrem<-function(betanew,data) {
  #We suposse the next structure in the dataset:Id (Identification of individual),..
# Individual mean (muki).
muki<-matrix(nrow=N,ncol=1)
muki[,1]<-exp(gamma0+as.vector(as.matrix(Dataprove[,6:(5+p)])%*%beta))

# Variance for the ARM.
muki2<-matrix(nrow=N,ncol=1)
muki2[,1]<-muki[,1]^2

# Finally, we define the variance.
Vk<-matrix(nrow=1,ncol=K)
inic<-1
end<-ngr[1]
for (i in 1:K) {Vk[,i]<-sigma2am*((muki[,i]^2)-(phik[,i]/(Dataprove[ini,5])))+(muki[,i]-phik[,i])*(1/(Dataprove[ini,5])))
inic<-1
end<-ngr[1+1]+end
}
sigma2amk<-sum(sigma2amk[1:K])/K

# First, we compute sigma square.

# Gradient.
Dkt<-t(Dk)
ElementKaGr<-list(matrix(nrow=K,ncol=1))
Gralist<-list(matrix(0,nrow=(p+1),ncol=1))
for (i in 1:K) {ElementKaGr[[i]]<-Dkt[,i]*(1/Vk[,i])*(Y[,i]-muk[,i])
Gralist[[1]]<-Gralist[[1]]+ElementKaGr[[i]]}
Gra<-matrix(nrow=(p+1),ncol=1)
for (i in 1:(p+1)) {Gra[,i]<-Gralist[[1]][i]}

# Hessian.
ElementkAhs<-list(matrix(nrow=K,ncol=1))
Hsalist<-list(matrix(0,nrow=(p+1),ncol=1))
for (i in 1:K) {ElementkAhs[[i]]<-1*matrix(Dkt[,i],nrow=(p+1),ncol=1)%*%(1/Vk[,i])%*%Dk[,i]
Hsalist[[1]]<-Hsalist[[1]]+ElementkAhs[[i]]}
Hsa<-matrix(,nrow=(p+1),ncol=(p+1))
for (i in 1:(p+1)){
  for (j in 1:(p+1)){Hsa[i,j]<-Hsalist[[1]][i,j]}}
#We return the Gradient (Gra), Hessian(Hsa) and sigma^2(sigm2am).
list(Gradienta=Gra,Hessiana=Hsa,sigma2am=sigm2am)}

########################################################.
########################################################.
##FUNCTION POPULATION-BASED ESTIMATING EQUATION (fpbm)##.
##FOR COMPUTE THE GRADIENT, HESSIAN AND SIGMAS ^ 2   ##.
########################################################.
########################################################.
fpbm<-function(betanew,data) {
#We suposse the next structure in the dataset:Id (Identification of individual) ,
#Group (Group number), Y (Individual outcome: 1 death, 0 alive), O (Population's.
#observed deaths), n (Population at risk), Z1 (covariate Z1), Z2 (covariate Z2) ,
#... ,Zp (covariate Zp).

Dataprove=data

K is the number of groups. We suppose that groups are ordered and they have all.
#The correlates numbers. For example:1,2,3 and not 1,3 (There are no number 2).
#N is the number of observations and p is the number of covariates.
N<-dim(Dataprove)[1]
K<-Dataprove[N,2]
p<-dim(Dataprove)[2]-5

ngr<-matrix(,nrow=1,ncol=K)
for (i in 1:K) ngr[1,i]<-dim(subset(Dataprove,Dataprove[2]==i))[1]
gamma0<-betanew[1]
beta<-betanew[-1]

###########################################.
#Gradient, Hessian for the individual part#.
###########################################.

#Individual outcome.
Yki<-matrix(,nrow=N,ncol=1)
Yki[,1]<-Dataprove[,3]

#Individual mean.
muki<-matrix(,nrow=N,ncol=1)
muki[,1]<-exp(gamma0+as.vector(as.matrix(Dataprove[,6:(5+p)])%*%beta))

#Individual matrix D.
Dki<-matrix(,nrow=N,ncol=p+1)
Dki[,1]<-muki[,1]
for (j in 1:p){
  Dki[,j+1]<-as.numeric(Dataprove[,5+j])*muki[,1]
}

#Inverse variance-covariance matrix individual part.
#First, we compute sigma square for the individual part.

muki2<-matrix(,nrow=N,ncol=1)
muki2[,1]<-muki[,1]^2

Yaver<-matrix(,nrow=1,ncol=K)
muk<-matrix(,nrow=1,ncol=K)
phik<-matrix(,nrow=K,ncol=1)
sigma2rek<-matrix(,nrow=K,ncol=1)
ini<-1
end<-ngr[1]
for (i in 1:K) {
  Yaver[1,i]<-sum(Yki[ini:end])/ngr[i]
  muk[1,i]<-sum(muki[ini:end])/ngr[i]
  phik[1,i]<-sum(muki2[ini:end])/ngr[i]
  sigma2rek[i,1]<-max((Yaver[1,i]*(Yaver[1,i]*ngr[i]-2*ngr[i]*muk[1,i]-
  1)+2*(t(muki[ini:end,1])%*%Yki[ini:end,1])/ng[i])/(ngr[1]*(muk[1,i]^2)-phik[1,i]+1,-100))
  ini<-end+1
  end<-ngr[i+1]+end}
sigma2re<-sum(sigma2rek[1:K])/K
#We compute the expression for one part (transpose(muk)*Inverse(Deltak)*muk).
sumk<-matrix(,nrow=1,ncol=K)
in<-1
end<-ngr[1]
for (i in 1:K) {sumk[1,i]<-sum((muki[ini:end]^2)/(muki[ini:end]*(1-(1+sigma2re)*muki[ini:end])))
  ini<--end+1
  end<--ngr[i+1]+end}

#Finally we define the elements of the inverse of v.
Vki<--list(matrix(nrow=K,ncol=1))
for (j in 1:K)
  Vki[[j]]<--matrix(0,nrow=ngri,j,ncol=ngri[j])
for (i in 1:ngri[j]) {yy<--1-(1+sigma2re)*muki[ini:end])
  Vki[[j]][i,i]<--(-sigma2re*(1/yy)*(1/(1-(1+sigma2re)*muki[ini:end]))
    Vki[[j]][i,(i:ngr[i+1])]<--(-sigma2re*(1/yy)*yy)
    Vki[[j]]=(1/(1+sigma2re*sumk[1,j]))
  }
#Vectors of individual responses for each group. For ngr[4] is NA but we don't use it.
Ykilist<--list(matrix(nrow=K,ncol=1))
#Vectors of mean for each group.
mukilist<--list(matrix(nrow=K,ncol=1))
ini<-1
end<--ngr[1]
for (i in 1:K) {Ykilist[i]<--list(matrix(Dataprove[ini:end,3],nrow=ngri[i],ncol=1))
  mukilist[i]<--list(matrix(muki[ini:end],nrow=ngri[i],ncol=1))
  ini<--end+1
  end<--ngr[i+1]+end}

#Vector diference response and mean.
Ykiminusmuki<--list(matrix(nrow=K,ncol=1))
for (i in 1:K) {Ykiminusmuki[i]<--Ykilist[i]-mukilist[i])}

#Matrix Dk for each group.
Dkilist<--list(matrix(nrow=K,ncol=1))
for (i in 1:K) Dkilist[i]<--matrix(nrow=ngri[1],ncol=p+1)
ini<-1
end<--ngr[1]
for (j in 1:K)
  for (n in 1:(p+1)){
    Dkilist[][n]<--Dki[ini:end,n]
    ini<--end+1
    end<--ngr[j+1]+end}

#Gradient Individual part.
ElementKGr<--list(matrix(nrow=K,ncol=1))
Grlist<--list(matrix(0,nrow=(p+1),ncol=1))
for (i in 1:K) \ElementKGr[[i]]<--t(Dkilist[i])*Vki[i]*Ykiminusmuki[i])
Grlist[[1]]<--Grlist[[1]]+ElementKGr[i]]

#Hessian individual part.
ElementKhs<--list(matrix(nrow=K,ncol=1))
Hslist<--list(matrix(0,nrow=(p+1),ncol=(p+1)))
for (i in 1:K) \ElementKhs[[i]]<--Hslist[i])*Vki[i]*Dkilist[i]]
Hslist[[1]]<--Hslist[[1]]+ElementKhs[i]]

#Outcome for the aggregated data model with combined analytical and aggregated models.
Ybar<--matrix(nrow=1,ncol=K)
#Matrix D for the aggregated part.
Dk<-matrix(nrow=K,ncol=p+1)
in<--1
end<--ngr[1]
for (i in 1:K) {Ybar[[i]]<--((Dataprove[ini,4]-sum(Yki[ini:end]))/(Dataprove[ini,5]-ngri[i]))
  for (j in 1:K) {Dk[i,j]<--sum(Dki[ini:end,j])/ngri[i]}}
ini<-end+1
dend<ngr[i+1]+end}

# Sigma square aggregated part.
sigma2pbk<-matrix(,nrow=K,ncol=1)
ini<-1
dend<ngr[1]
for (i in 1:K) {sigma2pbk[i,]<-max(((Ybar[,i]-muk[,i])^2-(muk[,i]-phik[,i]/(Dataprove[ini,5]-ngr[i])))^((muk[,i]^2-phik[,i]/(Dataprove[ini,5]-ngr[i]))),0)
ini<-end+1
dend<ngr[i+1]+end}

sigma2pb<-sum(sigma2pbk[1:K])/K

# Variance aggregated part.
Dkt<-t(Dk)
Vkbar<-matrix(,nrow=1,ncol=K)
ElementKarGr<-list(matrix(,nrow=K,ncol=1))
Grarlist<-list(matrix(0,nrow=(p+1),ncol=1))
ini<-1
dend<ngr[1]
for (i in 1:K) {Vkbar[1,i]<-sigma2pb*((muk[i]^2)-(phik[,i]/(Dataprove[ini,5]-ngr[i])))^((muk[i]^2-phik[,i]/(Dataprove[ini,5]-ngr[i])))
ElementKarGr[[i]]<-Dkt[,i]*(1/Vkbar[i])*(Ybar[i]-muk[i])
Grarlist[[1]]<-Grarlist[[1]]+ElementKarGr[[i]]
ini<-end+1
dend<ngr[i+1]+end}

# Gradient aggregated part.
Grar<-matrix(,nrow=(p+1),ncol=1)
for (i in 1:(p+1)) {Grar[i,1]<-Grarlist[[1]][i]}

# Hessian aggregated part.
ElementKarHs<-list(matrix(,nrow=K,ncol=1))
Hsarlist<-list(matrix(0,nrow=(p+1),ncol=(p+1)))
for (i in 1:K) {ElementKarHs[[i]]<--1*matrix(Dkt[,i],nrow=(p+1),ncol=1)%*%(1/Vkbar[i])%*%Dk[i,]
Hsarlist[[1]]<-Hsarlist[[1]]+ElementKarHs[[i]]
for (i in 1:K) {Hsar[i,j]<-Hsarlist[[1]][i,j]}

# Gradient, Hessian for the individual and aggregated part combination.
# Gradient PBEE.
Grpb<-c(Gr,Grar)
# Hessian PBEE.
Hspb<-c(Hs,Hsar)

# We return the gradient (Grpb), Hessian(Hspb), sigma^2 individual.
# part (sigma2re) and sigma^2 aggregated part (sigma2pb).
list(Gradientpb=Grpb,Hessianpb=Hspb,sigma2re=sigma2re,sigma2pb=sigma2pb)
if(sum(is.finite(q$Hessian))<Hdim) itercount=999
if(itercount>maxiter) break
betanew<-betanew-ginv(q$Hessian)%*%q$Gradient
itercount=itercount+1
if(is.na(q$sigma2re)) itercount=100
if(itercount>maxiter) break
if(q$sigma2re>50) itercount=100
if(itercount>maxiter) break
}
list(betanew=betanew,sigma2re=q$sigma2re, iterN=itercount)
}
fagrem2<-function(data,tol,maxiter=100,betainitial){
betanewa=betainitial
betaold<-betanewa+1
itercount=0
Hdim=length(betanewa)^2
#Solutions for the ARM.
while(max(abs((betaold-betanewa)/betaold))>tol){
  d<-fagrem(betanewa,data)
  betaold<-betanewa
  if(sum(is.finite(d$Hessiana))<Hdim) itercount=999
  if(itercount>maxiter) break
  betanewa<-betanewa-ginv(d$Hessiana)%*%d$Gradienta
  itercount=itercount+1
  if(is.na(d$sigma2am)) itercount=100
  if(itercount>maxiter) break
  if(d$sigma2am>50) itercount=100
  if(itercount>maxiter) break
}
list(betanewa=betanewa,sigma2am=d$sigma2am, iterN=itercount)
}
fpbm2<-function(data,tol,maxiter=100,betainitial){
betanewpb=betainitial
betaold<-betanewpb+1
itercount=0
Hdim=length(betanewpb)^2
#Solutions for the PBEE.
while(max(abs((betaold-betanewpb)/betaold))>tol){
  e<-fpbm(betanewpb,data)
  betaold<-betanewpb
  if(sum(is.finite(e$Hessianpb))<Hdim) itercount=999
  if(itercount>maxiter) break
  betanewpb<-betanewpb-ginv(e$Hessianpb)%*%e$Gradientpb
  itercount=itercount+1
  if(is.na(max(e$sigma2re,e$sigma2pb))) itercount=100
  if(itercount>maxiter) break
  if(max(e$sigma2re,e$sigma2pb)>50) itercount=100
  if(itercount>maxiter) break
}
list(betanewpb=betanewpb,sigma2re=e$sigma2re,sigma2pb=e$sigma2pb, iterN=itercount)
}
fgenerate2<-function(group,populationsize,samplesize,var1=.85,var2=4){
#group=number of groups, populationsize= population size.
#samplesize= sample size, var1 within group-variance X1Ki,.
#var2=within-group variance X2ki.
K<-group
nk<-populationsize
mk<-samplesize
#Covariate X1ki & X2ki. They are correlated 0.3 at the community.
#and individual levels (see Prentice & Sheppard).
covl=0.3*3.4*0.25
covm=matrix(c(3.4,covl,covl,0.25),2,2))
cov=0.3*sqrt(var1*var2)
covm=matrix(c(var1,cov,cov,var2),2,2)
x1ki=matrix(0,nk,K)
}
x2ki = matrix(0,nk,K)
for(i in 1:K){
  znk = mvrnorm(nk,zK[i,],covm)
  X1ki[,i] = znk[,1]
  X2ki[,i] = znk[,2]
}

# Country specific frailties were generated as independent.
# realized values from a gamma distribution with mean 1.
# and variance sigma^2. The mean of a gamma is shape*scale.
# and the variance is shape*(scale^2).

meanhk <- 1
varhk <- (meanhk^2)/(varhk)
shape <- (varhk)/meanhk
scale <- (varhk)/meanhk
hk <- rgamma(K,shape=shape,scale=scale)
hk = t(matrix(rep(hk,nk),nrow=K,ncol=nk))

# The disease events, yki, were generated by determining.
# wether a uniform random variable was less than.
# hk*exp(gamma0+beta1*X1ki+beta2*X2ki).

gamma0 <- -3
beta1 <- 0.2
beta2 <- 0.2
yki <- matrix(, nrow=nk, ncol=K)
unif <- matrix(runif(nk*K,0,1), nrow=nk, ncol=K)
yki = ifelse(unif < hk*exp(gamma0+beta1*X1ki+beta2*X2ki), 1, 0)

# selection of random sample of size mk#
# and organize data to apply functions #.
# farem, fagrem and fpbm

# selection of random sample of size mk#
# and organize data to apply functions #.
# farem, fagrem and fpbm

dataframe<- function(seed, Niter, K, N){
  set.seed(seed)
  count1 = 0
  result1 = matrix(0, nrow=Niter, ncol=16)
  while(count1 < Niter){
    tempdata1 = fgenerate2(K, 2000, N)
    tol = 0.001
    gg <- glm(YIND~X1ki+X2ki, data=tempdata1, family=binomial)
    beta1 <- as.vector(c(matrix(gg$coefficients[1]), matrix(gg$coefficients[2]), matrix(gg$coefficients[3])))
    count1 = count1 + 1
  }
  return(result1)
}

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#

#Simulation results#
#Simulation results#
tempdata1a1 <- farem2(tempdata1, tol, maxiter = 50, betaini)
tempdata1a2 <- fagrem2(tempdata1, tol, maxiter = 50, betaini)
tempdata1a3 <- fpsbm2(tempdata1, tol, maxiter = 50, betaini)

result1[count1, 1:5] = matrix(c(t(tempdata1a1$betanew), tempdata1a1$sigma2re, tempdata1a1$iterN), nrow = 1, ncol = 5)
result1[count1, 6:10] = matrix(c(t(tempdata1a2$betanewa), tempdata1a2$sigma2am, tempdata1a2$iterN), nrow = 1, ncol = 5)
result1[count1, 11:16] = matrix(c(t(tempdata1a3$betanewpb), tempdata1a3$sigma2re, tempdata1a3$sigma2pb, tempdata1a3$iterN), nrow = 1, ncol = 6)

print(count1)

list(result = result1, K = K, N = N)

#100 groups-100 sample size in each group.
finalresult100100 = fsimulationC(123, 1000, 100, 100)
save(list = c("finalresult100100", "Random.seed"), file = "100100C.RData")

#100 groups-50 sample size in each group.
finalresult10050 = fsimulationC(123, 1000, 100, 50)
save(list = c("finalresult10050", "Random.seed"), file = "10050C.RData")

#50 groups-100 sample size in each group.
finalresult50100 = fsimulationC(123, 1000, 50, 100)
save(list = c("finalresult50100", "Random.seed"), file = "50100C.RData")

#50 groups-50 sample size in each group.
finalresult5050 = fsimulationC(123, 1000, 50, 50)
save(list = c("finalresult5050", "Random.seed"), file = "5050C.RData")

A.3.4 NCC coverage program.

library(MASS)

#FUNCTION INDIVIDUAL RANDOM EFFECTS MODEL (faremcoverage)#.
#FOR COMPUTE NAIVE AND SANDWICH ESTIMATOR #.

faremcoverage <- function(betain, data) {
  betanew = as.vector(betain[1,], mode = "numeric")
  Dataprove = data
  K = dim(Dataprove)[2]
  N = dim(Dataprove)[1]
  p = dim(Dataprove)[2] - 5
  ngr = matrix(rnorm(N, mean = 1, nrow = K, ncol = K)
  for (i in 1:K) ngr[1, i] <- dim(subset(Dataprove, Dataprove[2] == i))[, 1]
  gamma0 = betanew[1]
  betanew[1]
  betanew <- betanew[-1]

  # Individual outcome.
  Yki = matrix(rnorm(N, mean = 1, nrow = N, ncol = 1)
  Yki[, 1] <- Dataprove[, 3]

  # Individual mean.
  muki = matrix(rnorm(N, mean = 1, nrow = N, ncol = 1)
  muki[, 1] <- exp(gamma0 + as.vector(as.matrix(Dataprove[, 6:(5 + p)]) %*% beta))

  # Matrix D for the IRM.
  DKi = matrix(rnorm(N, mean = 1, nrow = N, ncol = p + 1)
  DKi[, 1] <- muki[, 1]
  for (j in 1:p) {
    DKi[, j + 1] <- as.numeric(Dataprove[, 5 + j]) * muki[, 1]
  }

  # Inverse variance-covariance matrix for the IRM.
  muk12 = matrix(rnorm(N, mean = 1, nrow = N, ncol = 1)
  muk12[, 1] <- muki[, 1]^2
  Yaver = matrix(rnorm(N, mean = 1, nrow = 1, ncol = K)
  muk12 <- matrix(rnorm(1, mean = 1, nrow = 1, ncol = K)
}
\begin{verbatim}
phik <- matrix(, nrow = K, ncol = 1)
sigma2rek <- matrix(, nrow = K, ncol = 1)
in i <- 1
end <- ngr[1]
for (i in 1:K) {
    Yaver[1,i] <- sum(Yki[ini:end])/ngr[i]
muk[1,i] <- sum(muki[ini:end])/ngr[i]
phik[1,i] <- sum(muki2[ini:end])/ngr[i]
sigma2rek[1,i] <- max((Yaver[1,i] * ngr[i]-2*ngr[i] * muk[1,i]-1)+2*{(t(muki[ini:end,1])%*%Yki[ini:end,1])/ngr[1]})/(ngr[i] * (muk[1,i]^2)-phik[1,i])+1,-100)
    ini <- end+1
    end <- ngr[i+1]+end
}
sigma2re <- sum(sigma2rek[1:K])/K

# We compute the expression for one part (transpose(muk)*Inverse(Deltak)*muk).
sumk <- matrix(, nrow = 1, ncol = K)
in i <- 1
end <- ngr[1]
for (i in 1:K) {sumk[1,i] <- sum((muki[ini:end]^2)/(muki[ini:end]*(1-(1+sigma2re)*muki[ini:end])))
    ini <- end+1
    end <- ngr[i+1]+end}

# Finally, we define the elements of the inverse of V.

for (i in 1:K)
    Vki[i] <- matrix(0, nrow = ngr[1,i], ncol = ngr[1,i])
for (i in 1:K) {for (n in 1:(p+1))
    Dki[i,n] <- Dki[ini:end,n]
    ini <- end+1
    end <- ngr[i+1]+end}

for (i in 1:K) {ElementKM[i] <- t(Dki[i])%*%Vki[i]%*%Ykiminusmuki[i]%*%t(Ykiminusmuki[i])
    Mlist[i] <- matrix(0, nrow = (p+1), ncol = (p+1))
    for (i in 1:K)
        ElementKHs[i] <- t(Dki[i])%*%Ykiminusmuki[i] * t(Ykiminusmuki[i])
        Mlist[i] <- Mlist[i] + ElementKHs[i]
    M <- matrix(, nrow = (p+1), ncol = (p+1))
    for (i in 1:(p+1)) {for (j in 1:(p+1)) {M[i,j] <- Mlist[i][j]}}
    Hslist <- list(matrix(0, nrow = (p+1), ncol = (p+1)))
\end{verbatim}
for (i in 1:K) {
  ElementKHs[[i]] <- 1*t(Dkilist[[i]])%*%Vki[[i]]%*%Dkilist[[i]]
  Hslist[[1]] <- Hslist[[1]] + ElementKHs[[i]]
}

Hs <- matrix(nrow=(p+1),ncol=(p+1))
for (i in 1:(p+1)) {
  for (j in 1:(p+1)) {Hs[i,j] <- Hslist[[1]][i,j]}
}

naive <- ginv(-Hs)
robust <- naive%*%M%*%naive
list(naive=naive, robust=robust)
}

fagremcoverage <- function(betain, data) {
  betanew <- as.vector(betain[1,], mode="numeric")
  Dataprove = data
  #K is the number of groups. We suppose that groups are ordered and they have all.
  #the correlatives numbers. For example: 1, 2, 3 and not 1, 3 (There are no number 2).
  #N is the number of observations and p is the number of covariates.
  N <- dim(Dataprove)[1]
  K <- Dataprove[N,2]
  p <- dim(Dataprove)[2]-5
  ngr <- matrix(nrow=1, ncol=K)
  for (i in 1:K) ngr[1,i] <- dim(subset(Dataprove, Dataprove[2]==i))[1]
  gamma0 <- betanew[1]
  beta <- betanew[-1]

  #Individual mean (muki).
  muki <- matrix(nrow=N, ncol=1)
  muki[,1] <- exp(gamma0 + as.vector(as.matrix(Dataprove[,6:(5+p)])%*%beta))

  #Individual matrix D.
  Dki <- matrix(nrow=N, ncol=p+1)
  Dki[,1] <- muki[,1]
  for (j in 1:p) {Dki[,j+1] <- as.numeric(Dataprove[,5+j])*muki[,1]}

  #Variance for the ARM.
  muki2 <- matrix(nrow=N, ncol=1)
  muki2[,1] <- muki[,1]^2

  #Outcome for the ARM as defined in Sheppard and Prentice (Biometrics, 1995).
  Y <- matrix(nrow=1, ncol=K)
  muk <- matrix(nrow=1, ncol=K)
  phik <- matrix(nrow=K, ncol=1)

  #Matrix D for the ARM.
  Dk <- matrix(nrow=K, ncol=p+1)

  #First, we compute sigma square.
  sigma2amk <- matrix(nrow=1, ncol=K)
  ini <- 1
  end <- ngr[1]
  for (i in 1:K) {
    Y[1,i] <- ((Dataprove[ini,4])/(Dataprove[ini,5]))
    muki[1,i] <- sum(muki[ini:end])/ngr[i]
    phik[i,1] <- sum(muki[ini:end])/ngr[i]
    for (j in 1:(p+1)) {Dki[i,j] <- sum(Dki[ini:end,j])/ngr[i]}
    sigma2amk[[i]] <- max(((Y[1,i]-muki[i,1])^2-(muki[i,1]*Dk[,1,1]))/(muki[i,1]^2-phik[i,1]*(1/(Dataprove[ini,5])))), -100)
    ini <- end+1
    end <- ngr[i+1]+end}
  sigma2am <- sum(sigma2amk[1:K])/K

  #Finally, we define the variance.
  Vk <- matrix(nrow=1, ncol=K)
  ini <- 1
  end <- ngr[1]
  for (i in 1:K) {Vk[1,i] <- sigma2am*((muki[i,1]^2)-(phik[i,1]/(Dataprove[ini,5]))) + (muki[i,1]-phik[i,1])*(1/(Dataprove[ini,5]))}
  ini <- 1
  end <- ngr[1]
Dkt <- t(Dkt)
ElementKMa <- list(matrix(, nrow = K, ncol = 1))
Malist <- list(matrix(0, nrow = (p + 1), ncol = (p + 1)))
for (i in 1:K) {ElementKMa[[i]] <- matrix(Dkt[, i], (p + 1), 1) %*% matrix(Dkt[, i], 1, (p + 1)) * ((1/Vk[i]) * (Y[i] - muk[i]))^2
Malist[[1]] <- Malist[[1]] + ElementKMa[[i]]}
Ma <- matrix(, nrow = (p + 1), ncol = (p + 1))
for (i in 1:(p + 1)) {for (j in 1:(p + 1)) {Ma[i, j] <- Malist[[1]][i, j]}}
ElementKHa <- list(matrix(, nrow = K, ncol = 1))
Hsalist <- list(matrix(0, nrow = (p + 1), ncol = (p + 1)))
for (i in 1:K) {ElementKHa[[i]] <- -1 * matrix(Dkt[, i], nrow = (p + 1), ncol = 1) %*% (1/Vk[i]) %*% Dk[i, ]}
Hsalist[[1]] <- Hsalist[[1]] + ElementKHa[[i]]
Hsa <- matrix(, nrow = (p + 1), ncol = (p + 1))
for (i in 1:(p + 1)) {for (j in 1:(p + 1)) {Hsa[i, j] <- Hsalist[[1]][i, j]}}
naive <- ginv(-Hsa)
robust <- naive %*% Ma %*% naive
list(naive = naive, robust = robust)

dkt <- t(Dk)
ElementKMa <- list(matrix(, nrow = K, ncol = 1))
Malist <- list(matrix(0, nrow = (p + 1), ncol = (p + 1)))
for (i in 1:K) {ElementKMa[[i]] <- matrix((Dkt[i, ](p + 1)) * (%matrix((Dkt[i, ](p + 1) * (%(1/Vk[i]) * (Y[i] - muk[i])))^2
Malist[[1]] <- Malist[[1]] + ElementKMa[[i]]}
Ma <- matrix(, nrow = (p + 1), ncol = (p + 1))
for (i in 1:(p + 1)) {for (j in 1:(p + 1)) {Ma[i, j] <- Malist[[1]][i, j]}}
ElementKHa <- list(matrix(, nrow = K, ncol = 1))
Hsalist <- list(matrix(0, nrow = (p + 1), ncol = (p + 1)))
for (i in 1:K) {ElementKHa[[i]] <- -1 * matrix((Dkt[i, ](p + 1), ncol = 1) %*% %%(1/Vk[i]) %*% Dk[i, ]}
Hsalist[[1]] <- Hsalist[[1]] + ElementKHa[[i]]
Hsa <- matrix(, nrow = (p + 1), ncol = (p + 1))
for (i in 1:(p + 1)) {for (j in 1:(p + 1)) {Hsa[i, j] <- Hsalist[[1]][i, j]}}
naive <- ginv(-Hsa)
robust <- naive %*% Ma %*% naive
list(naive = naive, robust = robust)

fpbmcoverage <- function(betain, data) {
betanew = as.vector(betain[1, ], mode = "numeric")
Dataprove = data
# K is the number of groups. We suppose that groups are ordered and they have all.
# K the correlatives for example: 1, 2, 3 and not 1, 3 (there are no number 2).
# N is the number of observations and p is the number of covariates.
N <- dim(Dataprove)[1]
K <- ncol(Dataprove) - 5
ngr <- matrix(, nrow = 1, ncol = K)
for (i in 1:K) {ngr[1, i] <- dim(subset(Dataprove, Dataprove[2] == i))[1]}
gamma0 <- betanew[1]
beta <- betanew[-1]

# Individual outcome.
Yki <- matrix(, nrow = N, ncol = 1)
Yki[, 1] <- data[, 3]

# Individual mean.
muki <- matrix(, nrow = N, ncol = 1)
muki[, 1] <- exp(gamma0 + as.vector(as.matrix(data[6:5+p])) %*% beta)

# Individual matrix D.
Dki <- matrix(, nrow = N, ncol = p + 1)
Dki[, 1] <- muki[, 1]
for (j in 1:p) {Dki[, j + 1] <- as.numeric(data[5 + j]) * muki[, 1]}
Yaver[1,i]<-sum(Yki[ini:end])/ngr[i]
muk[1,i]<-sum(muki[ini:end])/ngr[i]
phik[1,i]<-sum(muki2[ini:end])/ngr[i]
sigma2rek[1,i]<-max(((Yaver[1,i]*Yaver[1,i]-2*ngr[i]*muk[1,i]-1)+2*((muki[ini:end,1])%*%Yki[ini:end,1])/ngr[i]))/(ngr[i]*(muk[1,i]^2)-phik[1,i]+1,-100)
ini<-end+1
end<-ngr[i+1]+end

sigma2rek<-sum(sigma2rek[1:K])/K

#Finally we define the elements of the inverse of V.
Vki<-list(matrix(,nrow=K,ncol=1))
for (j in 1:K){
  Vki[[j]]<-matrix(0,nrow=ngr[1,j],ncol=ngr[1,j])
  for (i in 1:ngr[1,j]) {
    yy=1/(1+sigma2re^*muki[i])
    Vki[[j]][[1+(1:j)]]<-(-sigma2re*(1/yy)*(1/(1-(1+sigma2re^*muki[i])))^2)/(1+sigma2re^*sumk[i,j],j])
    Vki[[j]][i,j]<-(1/(muki[i]*yy))-
    (sigma2re*(1/yy)^2)/(1/(1+sigma2re^*sumk[i,j],j])
    Vki[[j]][i,j]<-(Vki[[j]])+(Vki[[j]])-diag(diag(Vki[[j]]))
  }
}

#Vectors of individual responses for each group. For ngr[4] is NA but we don't use it.
Ykilist<-list(matrix(,nrow=K,ncol=1))

#Vectors of mean for each group.
mukilist<-list(matrix(,nrow=K,ncol=1))
ini<-1
end<-ngr[1]
for (i in 1:K) {Ykilist[i]<-list(matrix(Dataprove[ini:end,3],nrow=ngr[i],ncol=1))
mukilist[i]<-list(matrix(muki[ini:end],nrow=ngr[i],ncol=1))
ini<-end+1
end<-ngr[i+1]+end}

#Vector difference response and mean.
Ykiminusmuki<-list(matrix(,nrow=K,ncol=1))

for (i in 1:K) {Ykiminusmuki[i]<-(Ykilist[i]-mukilist[i])

#Matrix Dk for each group.
Dkilist<-list(matrix(,nrow=K,ncol=1))
for (i in 1:K) Dkilist[i]<-matrix(,nrow=ngr[i],ncol=p+1)
ini<-1
end<-ngr[1]
for (j in 1:K){
  for (n in 1:(p+1)){
    Dkilist[i][j,n]<-Dki[ini:end,n]
    ini<-end+1
    end<-ngr[j+1]+end
  }
}

#Outcome for the aggregating data model with combined analytical and aggregated models.
Ybar<-matrix(,nrow=1,ncol=K)

#Matrix D for the aggregated part.
Dk<-matrix(,nrow=K,ncol=p+1)
ini<-1
end<-ngr[1]
for (i in 1:K) {Ybar[1,i]<-((Dataprove[ini,4]-sum(Yki[ini:end]))/(Dataprove[ini,5]-ngr[i]))

  for (j in 1:(p+1)){Dk[i,j]<-sum(Dki[ini:end,j])/ngr[i]
    ini<-end+1
    end<-ngr[i+1]+end
  }
}

Dkt<-t(Dk)

#Sigma square aggregated part.
sigma2pbk<-matrix(,nrow=K,ncol=1)
ini<-1
end<-ngr[1]
for (i in 1:K) {sigma2pbk[i]<-max(((Ybar[1,i]*muk[i,1]-phik[i])/(Dataprove[ini,5]-ngr[i]))/(muk[i,1]^2-phik[i]+1/(Dataprove[ini,5]-ngr[i]),0)
  ini<-end+1
  end<-ngr[i+1]+end
}
\[
\sigma_{pb} = \frac{\text{sum}(\sigma_{pb}[1:K])}{K}
\]
\[
V_{kbar} = \text{matrix}(0, \text{nrow}=(p+1), \text{ncol}=(p+1))
\]
\[
i=1 \to \text{end} \rightarrow \text{ngr}[i]
\]
\[
\text{for } (i \in 1:K) \{ \\
\quad V_{kbar}[1,i] = \sigma_{pb}^2 \left( \frac{\text{muk}[i]^2}{(\text{Dataprove}[ini,5] - \text{ngr}[i])} + \text{muk}[i] - \text{phik}[i] \right) \left( \frac{1}{(\text{Dataprove}[ini,5] - \text{ngr}[i])} \right) \\
\quad \text{junkmat} = \text{diag}(\text{rep}(0, \text{ngr}[i]), 1/V_{kbar}[1,i]) \\
\quad \text{junkmat}[1,1] = V_{kbar}[1,i] \\
\quad \text{sub} = \begin{bmatrix} \text{rbind}(\text{Oklist}[i], \text{Ok}[i,]) & \% \% \text{bind}(\text{Ykminusmuk}[i,], \text{Ybar}[i] - \text{muk}[i]) \end{bmatrix} \\
\quad \text{Mpb} = \text{Mpb} + \text{sub} \times \text{t(sub)} \\
\quad ini = \text{end} + 1 \\
\quad end = \text{ngr}[i+1] + end 
\}
\]
\[
\text{ElementKHs} = \text{list}(\text{matrix}(0, \text{nrow}=(p+1), \text{ncol}=(p+1)))
\]
\[
\text{for } (i \in 1:K) \{ \\
\quad \text{ElementKHs}[i] = -1 \times \text{Dkilist}[i] \times \text{Vki}[i] \times \text{Dkilist}[i] \\
\quad \text{Hslist}[i] = \text{Hslist}[i] + \text{ElementKHs}[i] 
\}
\]
\[
\text{Hs} = \text{matrix}(0, \text{nrow}=(p+1), \text{ncol}=(p+1))
\]
\[
\text{for } (i \in 1:1) \{ \\
\quad \text{for } (j \in 1:1) \{ \text{Hs}[i,j] = \text{Hslist}[i][i,j] \} 
\}
\]
\[
\text{ElementKarHs} = \text{list}(\text{matrix}(0, \text{nrow}=(p+1), \text{ncol}=(p+1)))
\]
\[
\text{for } (i \in 1:K) \{ \\
\quad \text{ElementKarHs}[i] = -1 \times \text{matrix}(\text{Dkt}[,i], \text{nrow}=(p+1), \text{ncol}=1) \times \frac{1}{V_{kbar}[i]} \times \text{Dk}[i,] \\
\quad \text{Hsarlist}[i] = \text{Hsarlist}[i] + \text{ElementKarHs}[i] 
\}
\]
\[
\text{Hsar} = \text{matrix}(0, \text{nrow}=(p+1), \text{ncol}=(p+1))
\]
\[
\text{for } (i \in 1:1) \{ \\
\quad \text{for } (j \in 1:1) \{ \text{Hsar}[i,j] = \text{Hsarlist}[i][i,j] \} 
\}
\]
\[
\text{Hspb} = \text{Hs} + \text{Hsar}
\]
\[
\text{naive} = \text{ginv}(-\text{Hspb}) \\
\text{robust} = \text{naive} \times \text{Mpb} \times \text{naive}
\]
\]

### Confidence Interval Naive and Sandwich for the IRM
### (function farem2coverage, ARM (function fagrem2coverage) AND PBEE (function fpbm2coverage)###

\[
\text{farem2coverage} = \text{function(data, datavar, beta1 = .2)} \{
\quad \text{result} = \text{rep(NA, 21)} \\
\quad \text{result}[1] = \text{datavar}[2] \\
\quad \text{if } (\text{datavar}[5] < 51) \{ \\
\quad \quad q = \text{faremcoverage(datavar[1:3], data)} \\
\quad \quad \text{naive} = q \times \text{naive} \\
\quad \quad \text{robust} = \text{naive} \times \text{Mpb} \times \text{naive} \\
\quad \quad \text{if } (\text{naive}[2,2] < 0) \{ \\
\quad \quad \quad \text{result}[2:3] = \text{rep(-1, 2)} \\
\quad \quad \quad \text{result}[16] = -1 \\
\quad \quad \text{if } (\text{robust}[2,2] < 0) \{ \\
\quad \quad \quad \text{result}[4:5] = \text{rep(-1, 2)} \\
\quad \quad \quad \text{result}[17] = -1 \\
\quad \quad \text{if } (\text{naive}[2,2] > 0) \{ \\
\quad \quad \quad \text{LL} = \text{datavar}[2] - 1.96 \times \text{sqrt(naive}[2,2]) \\
\quad \quad \quad \text{UL} = \text{datavar}[2] + 1.96 \times \text{sqrt(naive}[2,2]) \\
\quad \quad \quad \text{result}[2:3] = c(\text{LL, UL}) \\
\quad \quad \quad \text{result}[16] = \text{ifelse(betal} > \text{LL & betal <= UL, 1, 0)} \\
\quad \quad \text{if } (\text{robust}[2,2] = 0) \{ \\
\quad \quad \quad \text{LL} = \text{datavar}[2] - 1.96 \times \text{sqrt(robust}[2,2]) \\
\quad \quad \quad \text{UL} = \text{datavar}[2] + 1.96 \times \text{sqrt(robust}[2,2]) \\
\quad \quad \quad \text{result}[4:5] = c(\text{LL, UL}) \\
\quad \quad \quad \text{result}[17] = \text{ifelse(betal} > \text{LL & betal <= UL, 1, 0)} 
\quad \}
\quad \}
\quad \}
\}\n\]
else {
  result[c(2:5,16:17)]=rep(-1,6)
}

unlist(result)

fagrem2coverage<-function(data,datavar,beta1=.2){
  result=rep(NA,21)
  if (datavar[10]<51) {
    d<-fagremcoverage(datavar[6:8],data)
    naive=d$naive
    robust=d$robust
    if (naive[2,2]<0){
      result[7:8]=rep(-1,2)
      result[18]=-1
    }
    if (robust[2,2]<0){
      result[9:10]=rep(-1,2)
      result[19]=-1
    }
    if (naive[2,2]>=0){
      ll=(datavar[7]-1.96*sqrt(naive[2,2]))
      ul=(datavar[7]+1.96*sqrt(naive[2,2]))
      result[7:8]=c(ll,ul)
      result[18]=ifelse(beta1>=ll & beta1<=ul, 1, 0)
    }
    if (robust[2,2]>=0){
      ll=(datavar[7]-1.96*sqrt(robust[2,2]))
      ul=(datavar[7]+1.96*sqrt(robust[2,2]))
      result[9:10]=c(ll,ul)
      result[19]=ifelse(beta1>=ll & beta1<=ul, 1, 0)
    }
  }
  else {
    result[c(7:10,18:19)]=rep(-1,6)
  }
  unlist(result)
}

fpbm2coverage<-function(data,datavar,beta1=.2){
  result=rep(NA,21)
  if (datavar[16]<51) {
    pb<-fpbmcoverage(datavar[11:13],data)
    naive=pb$naive
    robust=pb$robust
    if (naive[2,2]<0){
      result[12:13]=rep(-1,2)
      result[20]=-1
    }
    if (robust[2,2]<0){
      result[14:15]=rep(-1,2)
      result[21]=-1
    }
    if (naive[2,2]>=0){
      ll=(datavar[12]-1.96*sqrt(naive[2,2]))
      ul=(datavar[12]+1.96*sqrt(naive[2,2]))
      result[12:13]=c(ll,ul)
      result[20]=ifelse(beta1>=ll & beta1<=ul, 1, 0)
    }
    if (robust[2,2]>=0){
      ll=(datavar[12]-1.96*sqrt(robust[2,2]))
      ul=(datavar[12]+1.96*sqrt(robust[2,2]))
      result[14:15]=c(ll,ul)
      result[21]=ifelse(beta1>=ll & beta1<=ul, 1, 0)
    }
  }
  else {
    result[c(12:15,20:21)]=rep(-1,6)
  }
}
unlist(result)

fgenerate2<-function(group, populationsize, samplesize, variance){
  # group = number of groups, populationsize = population size.
  # samplesize = sample size, variance = within group variance.

  K<-group
  nk<-populationsize
  mk<-samplesize
  varwithin<-variance

  # Covariate X1ki with ratio (within variance)/(between variance) equal varwithin/1.
  Z1kg<-rnorm(K,0,sqrt(1))
  X1ki<-t(matrix(rnorm(nk*K,Z1kg,sqrt(varwithin)),nrow=K,ncol=nk))

  # Covariate X2ki with ratio (within variance)/(between variance) equal 1/1.
  Z2kg<-matrix(rnorm(K,0,sqrt(1)),nrow=K,ncol=1)
  X2ki<-t(matrix(rnorm(nk*K,Z2kg,sqrt(1)),nrow=K,ncol=nk))

  # Country specific frailties were generated as independent.
  # real values from a gamma distribution with mean 1.
  # and variance sigma^2. The mean of a gamma is shape*scale.
  meanhk<-1
  varhk<-0.05
  shape<-(meanhk^2)/(varhk)
  scale<-(varhk)/(meanhk)
  hk<-rgamma(K,shape=shape,scale=scale)
  hk=t(matrix(rep(hk,nk),nrow=K,ncol=nk))

  # The disease events, yki, were generated by determining.
  # wether a uniform random variable was less than.
  gamma0<--3
  beta1<-0.2
  beta2<-0.2
  yki<-matrix(,nrow=nk,ncol=K)
  unif<-matrix(runif(nk*K,0,1),nrow=nk,ncol=K)
  yki<-(ifelse(unif<hk*exp(gamma0+beta1*X1ki+beta2*X2ki),1,0)

  #Selection of random sample of size mk#.
  # and organize data to apply functions #.

  datalist<-list(matrix(,nrow=K,ncol=1))
  sampledatalist<-list(matrix(,nrow=K,ncol=1))
  ini<-1
  end<-mk
  data<-matrix(,nrow=mk*K,ncol=5)
  for (i in 1:K){
    datalist[[i]]<-cbind(matrix(c(1:nk),nrow=nk,ncol=1),matrix(yki[,i],nrow=nk,ncol=1),matrix(X1ki[,i],nrow=nk,ncol=1),matrix(X2ki[,i],nrow=nk,ncol=1),matrix(c(i),nrow=nk,ncol=1))
    sampledatalist[[i]]<-(sample(datalist[[i]][,1],nrow=mk,ncol=1))
    data[ini:end,]<-(matrix(sample(datalist[[i]][,1],nrow=mk,ncol=1))
    ini<-end+1
    end<-mk*(i+1)
  }

  O<-matrix(apply(yki,2,sum),nrow=K,ncol=1)
  ini<-1
  end<-mk
  datapop<-matrix(,nrow=mk*K,ncol=1)
  for (i in 1:K) { datapop[ini:end,]<-(O[i,]
  ini<-end+1

  }
end<-mk*(i+1)}
datadatapop<-cbind(data,datapop,c(nk))
datafin<-data.frame(id=matrix(datadatapop[,1]),group=matrix(datadatapop[,5]),YIND=matrix(datadatapop[,2]),O=matrix(datadatapop[,6]),n=matrix(datadatapop[,7]),X1ki=matrix(datadatapop[,3]),X2ki=matrix(datadatapop[,4]))

###################.
##Coverage results##.
###################.
fsimulationAcoverage<-function(seed,Niter,sigma2,K,N,datavar){
  set.seed(seed)
count1=0
  result1=matrix(NA,Niter,21)
  while(count1<Niter){
    tempdata1=fgenerate2(K,2000,N,sigma2)
count1=count1+1
    a1<-farem2coverage(tempdata1,datavar[count1,])
a2<-fagrem2coverage(tempdata1,datavar[count1,])
a3<-fpbm2coverage(tempdata1,datavar[count1,])
    result1[count1,c(1:5,16:17)]=a1[c(1:5,16:17)]
    result1[count1,c(6:10,18:19)]=a2[c(6:10,18:19)]
    result1[count1,c(11:15,20:21)]=a3[c(11:15,20:21)]
    print(count1)
  }
  list(result=result1,sigma2=sigma2,K=K,N=N)
}

##########################################.
#100 groups-100 sample size in each group#.
##########################################.
#The next text files are the results files from the simulation runs.
#To obtain parameter estimates in each variation ratio for the 100-100 case.
#For example, dataAlvar1.txt is from finalresultA100100A.25$result.
dataAlvar1c=read.table("dataAlvar1.txt",header=T) #Variance 0.25.
dataAlvar2c=read.table("dataAlvar2.txt",header=T) #Variance 0.5.
dataAlvar3c=read.table("dataAlvar3.txt",header=T) #Variance 1.
dataAlvar4c=read.table("dataAlvar4.txt",header=T) #Variance 2.
dataAlvar5c=read.table("dataAlvar5.txt",header=T) #Variance 4.
dataAlvar6c=read.table("dataAlvar6.txt",header=T) #Variance 8.
dataAlvar7c=read.table("dataAlvar7.txt",header=T) #Variance 16.
coverage100100A.25=fsimulationAcoverage(123,1000,.25,100,100,dataAlvar1)
save(list=c("coverage100100A.25",".Random.seed"),file="100100coverageA025.RData")
savedseed=.Random.seed
coverage100100A.5=fsimulationAcoverage(savedseed,1000,.5,100,100,dataAlvar2)
save(list=c("coverage100100A.5",".Random.seed"),file="100100coverageA05.RData")
savedseed=.Random.seed
coverage100100A1=fsimulationAcoverage(savedseed,1000,1,100,100,dataAlvar3)
save(list=c("coverage100100A1",".Random.seed"),file="100100coverageA01.RData")
savedseed=.Random.seed
coverage100100A2=fsimulationAcoverage(savedseed,1000,2,100,100,dataAlvar4)
save(list=c("coverage100100A2",".Random.seed"),file="100100coverageA02.RData")
savedseed=.Random.seed
coverage100100A4=fsimulationAcoverage(savedseed,1000,4,100,100,dataAlvar5)
save(list=c("coverage100100A4",".Random.seed"),file="100100coverageA04.RData")
savedseed=.Random.seed
coverage100100A8=fsimulationAcoverage(savedseed,1000,8,100,100,dataAlvar6)
save(list=c("coverage100100A8",".Random.seed"),file="100100coverageA08.RData")
savedseed=.Random.seed
coverage100100A16=fsimulationAcoverage(savedseed,1000,16,100,100,dataAlvar7)
save(list=c("coverage100100A16",".Random.seed"),file="100100coverageA016.RData")

##########################################.
#50 groups-100 sample size in each group #.
##########################################.
#The next text files are the results files from the simulation runs.
#To obtain parameter estimates in each variation ratio for the 50-100 case.
dataA2var1c=read.table("dataA2var1.txt",header=T) #Variance 0.25.
dataA2var2c=read.table("dataA2var2.txt",header=T) #Variance 0.5.
dataA2var3c=read.table("dataA2var3.txt",header=T) #Variance 1.
dataA2var4<-read.table("dataA2var4.txt",header=T) #Variance 2.
dataA2var5<-read.table("dataA2var5.txt",header=T) #Variance 4.
dataA2var6<-read.table("dataA2var6.txt",header=T) #Variance 8.
dataA2var7<-read.table("dataA2var7.txt",header=T) #Variance 16.

coverage50100A.25=fsimulationAcoverage(123,1000, .25, 50, 100, dataA2var1)
save(list=c("coverage50100A.25", ",.Random.seed"), file="50100coverageA025.RData")
savedseed=.Random.seed

coverage50100A.5=fsimulationAcoverage(savedseed,1000,.5,50,100,dataA2var2)
save(list=c("coverage50100A.5",",.Random.seed"), file="50100coverageA05.RData")
savedseed=.Random.seed

coverage50100A1=fsimulationAcoverage(savedseed,1000,1,50,100,dataA2var3)
save(list=c("coverage50100A1",",.Random.seed"), file="50100coverageA1.RData")
savedseed=.Random.seed

coverage50100A2=fsimulationAcoverage(savedseed,1000,2,50,100,dataA2var4)
save(list=c("coverage50100A2",",.Random.seed"), file="50100coverageA2.RData")
savedseed=.Random.seed

coverage50100A4=fsimulationAcoverage(savedseed,1000,4,50,100,dataA2var5)
save(list=c("coverage50100A4",",.Random.seed"), file="50100coverageA4.RData")
savedseed=.Random.seed

coverage50100A8=fsimulationAcoverage(savedseed,1000,8,50,100,dataA2var6)
save(list=c("coverage50100A8",",.Random.seed"), file="50100coverageA8.RData")
savedseed=.Random.seed

coverage50100A16=fsimulationAcoverage(savedseed,1000,16,50,100,dataA2var7)
save(list=c("coverage50100A16",",.Random.seed"), file="50100coverageA16.RData")

#########################################.
#100 groups-50 sample size in each group#.
#########################################

#The next text files are the results files from the simulation runs.
#to obtain parameter estimates in each variation ratio for the 50-100 case.
dataA3var1<-read.table("dataA3var1.txt",header=T) #Variance 0.25.
dataA3var2<-read.table("dataA3var2.txt",header=T) #Variance 0.5.
dataA3var3<-read.table("dataA3var3.txt",header=T) #Variance 1.
dataA3var4<-read.table("dataA3var4.txt",header=T) #Variance 2.
dataA3var5<-read.table("dataA3var5.txt",header=T) #Variance 4.
dataA3var6<-read.table("dataA3var6.txt",header=T) #Variance 8.
dataA3var7<-read.table("dataA3var7.txt",header=T) #Variance 16.

coverage10050A.25=fsimulationAcoverage(123,1000, .25, 100, 50, dataA3var1)
save(list=c("coverage10050A.25",",.Random.seed"), file="10050coverageA025.RData")
savedseed=.Random.seed

coverage10050A.5=fsimulationAcoverage(savedseed,1000,.5,100,50,dataA3var2)
save(list=c("coverage10050A.5",",.Random.seed"), file="10050coverageA05.RData")
savedseed=.Random.seed

coverage10050A1=fsimulationAcoverage(savedseed,1000,1,100,50,dataA3var3)
save(list=c("coverage10050A1",",.Random.seed"), file="10050coverageA1.RData")
savedseed=.Random.seed

coverage10050A2=fsimulationAcoverage(savedseed,1000,2,100,50,dataA3var4)
save(list=c("coverage10050A2",",.Random.seed"), file="10050coverageA2.RData")
savedseed=.Random.seed

coverage10050A4=fsimulationAcoverage(savedseed,1000,4,100,50,dataA3var5)
save(list=c("coverage10050A4",",.Random.seed"), file="10050coverageA4.RData")
savedseed=.Random.seed

coverage10050A8=fsimulationAcoverage(savedseed,1000,8,100,50,dataA3var6)
save(list=c("coverage10050A8",",.Random.seed"), file="10050coverageA8.RData")
savedseed=.Random.seed

coverage10050A16=fsimulationAcoverage(savedseed,1000,16,100,50,dataA3var7)
save(list=c("coverage10050A16",",.Random.seed"), file="10050coverageA16.RData")

#########################################.
#50 groups-50 sample size in each group#.
#########################################

#The next text files are the results files from the simulation runs.
#to obtain parameter estimates in each variation ratio for the 50-100 case.
dataA4var1<-read.table("dataA4var1.txt",header=T) #Variance 0.25.
dataA4var2<-read.table("dataA4var2.txt",header=T) #Variance 0.5.
dataA4var3<-read.table("dataA4var3.txt",header=T) #Variance 1.
dataA4var4<-read.table("dataA4var4.txt",header=T) #Variance 2.
dataA4var5<-read.table("dataA4var5.txt",header=T) #Variance 4.
dataA4var6<-read.table("dataA4var6.txt",header=T) #Variance 8.
dataA4var7<-read.table("dataA4var7.txt",header=T) #Variance 16.

coverage5050A.25=fsimulationAcoverage(123,1000,.25,50,50,dataA4var1)
save(list=c("coverage5050A.25",".Random.seed"),file="5050coverageA025.RData")
savedseed=.Random.seed

coverage5050A.5=fsimulationAcoverage(savedseed,1000,.5,50,50,dataA4var2)
save(list=c("coverage5050A.5",".Random.seed"),file="5050coverageA05.RData")
savedseed=.Random.seed

coverage5050A1=fsimulationAcoverage(savedseed,1000,1,50,50,dataA4var3)
save(list=c("coverage5050A1",".Random.seed"),file="5050coverageA1.RData")
savedseed=.Random.seed

coverage5050A2=fsimulationAcoverage(savedseed,1000,2,50,50,dataA4var4)
save(list=c("coverage5050A2",".Random.seed"),file="5050coverageA2.RData")
savedseed=.Random.seed

coverage5050A4=fsimulationAcoverage(savedseed,1000,4,50,50,dataA4var5)
save(list=c("coverage5050A4",".Random.seed"),file="5050coverageA4.RData")
savedseed=.Random.seed

coverage5050A8=fsimulationAcoverage(savedseed,1000,8,50,50,dataA4var6)
save(list=c("coverage5050A8",".Random.seed"),file="5050coverageA8.RData")
savedseed=.Random.seed

coverage5050A16=fsimulationAcoverage(savedseed,1000,16,50,50,dataA4var7)
save(list=c("coverage5050A16",".Random.seed"),file="5050coverageA16.RData")

###################################################.
###COVERAGE INTERVAL OF THE ESTIMATES ##.
###################################################.

covinter<-function(result,Niter){
#result is the file with the estimate parameter b1 and the confidence
#interval with the naive estimator and sandwich in the IRM, ARM & PBEE.

matrixone=matrix(1,nrow=Niter,ncol=1)
resultarem=matrix(result[,16:17],ncol=2)
resultarem2=cbind(resultarem,matrixone)
resultagrem=matrix(result[,18:19],ncol=2)
resultagrem2=cbind(resultagrem,matrixone)
resultpbm=matrix(result[,20:21],ncol=2)
resultpbm2=cbind(resultpbm,matrixone)

resultaremsubnaive=matrix(subset(resultarem2,resultarem2[,1]>=0),ncol=3)
resultagremsubnaive=matrix(subset(resultagrem2,resultagrem2[,1]>=0),ncol=3)
resultpbmsubnaive=matrix(subset(resultpbm2,resultpbm2[,1]>=0),ncol=3)

resultaremsubsandwich=matrix(subset(resultarem2,resultarem2[,2]>=0),ncol=3)
resultagremsubsandwich=matrix(subset(resultagrem2,resultagrem2[,2]>=0),ncol=3)
resultpbmsubsandwich=matrix(subset(resultpbm2,resultpbm2[,2]>=0),ncol=3)

sumaremsnaive=sum(resultaremsubnaive[,1])
naremsnaive=sum(resultaremsubnaive[,3])
sumagremsnaive=sum(resultagremsubnaive[,1])
nagremsnaive=sum(resultagremsubnaive[,3])
sumpbmsnaive=sum(resultpbmsubnaive[,1])
npbmsnaive=sum(resultpbmsubnaive[,3])

sumaremsubsandwich=sum(resultaremsubsandwich[,2])
naremsubsandwich=sum(resultaremsubsandwich[,3])
sumagremsubsandwich=sum(resultagremsubsandwich[,2])
nagremsubsandwich=sum(resultagremsubsandwich[,3])
sumpbmsubsandwich=sum(resultpbmsubsandwich[,2])
npbmsubsandwich=sum(resultpbmsubsandwich[,3])

#coverage interval for IRM.
aremnaive=sumaremsnaive/naremsnaive
aremsubsandwich=sumaremsubsandwich/naremsubsandwich

#coverage interval for ARM.
agremsnaive=sumagremsnaive/nagremsnaive
agremsubsandwich=sumagremsubsandwich/nagremsubsandwich

#coverage interval for PBEE.
pbmsnaive=sumpbmsnaive/npbmsnaive
pbmsubsandwich=sumpbmsubsandwich/npbmsubsandwich


cat("Coverage interval naive
(AREM,AGREM,PBM)","\n",aremnaive,"\n",agremsnaive,"\n",pbmsnaive,"\n")

219
A.3.5 SCC coverage program.

```r
library(MASS)

# Individual random effects model (faremcoverage)#
# FOR COMPUTE NAIVE AND SANDWICH ESTIMATOR #######
# library(MASS)
# Individual random effects model (faremcoverage)####
# FOR COMPUTE NAIVE AND SANDWICH ESTIMATOR #######

faremcoverage <- function(betain, data) {
  betanew <- as.vector(betain[, 1], mode = "numeric")
  Dataprove <- data

  K <- dim(Dataprove)[2]
  p <- dim(Dataprove)[2] - 5

  ngr <- matrix(, nrow = 1, ncol = K)
  for (i in 1:K) ngr[1, i] <- dim(subset(Dataprove, Dataprove[, 2] == i))[1]

  gamma0 <- betanew[1]
  beta <- betanew[-1]

  # Individual outcome.
  Yki <- matrix(, nrow = N, ncol = 1)
  Yki[, 1] <- Dataprove[, 3]

  # Individual mean.
  muki <- matrix(, nrow = N, ncol = 1)
  muki[, 1] <- exp(gamma0 + as.vector(as.matrix(Dataprove[, 6:(5 + p)]) %*% beta))

  # Matrix D for the IRM.
  Dki <- matrix(, nrow = N, ncol = p + 1)
  Dki[, 1] <- muki[, 1]
}
```

for (j in 1:p){
  Dki[,j+1]<-as.numeric(Dataprove[,5+j])*muki[,1]
}

#Inverse variance-covariance matrix for the IRM.

##First, we compute sigma_square.
muki2<-matrix(,nrow=N,ncol=1)
muki2[,1]<-muki[,1]^2

Yaver<-matrix(,nrow=1,ncol=K)
muk<-matrix(,nrow=K,ncol=1)
phik<-matrix(,nrow=K,ncol=1)
sigma2rek<-matrix(,nrow=K,ncol=1)
inici-1
endi<-ngr[1]
for (i in 1:K) {
  Yaver[1,i]<-sum(Yki[ini:end])/ngr[i]
  muk[1,i]<-sum(muki[ini:end])/ngr[i]
  phik[1,i]<-sum(muki2[ini:end])/ngr[i]
  sigma2rek[1,i]<-max((Yaver[1,i]*(Yaver[1,i]*ngr[i]-2*ngr[i]*muk[1,i]-
  1)+2*((t(muki[ini:end,1])%*%Yki[ini:end,1])/ngr[i])*(muk[1,i]^2)-phik[1,i]+1,-
  100)
  inici<end
  endi<-ngr[i+1]+end
}
sigma2re<-sum(sigma2rek[1:K])/K

#We compute the expression for one part (transpose(muk)*Inverse(Deltak)*muk).

sumk<-matrix(,nrow=1,ncol=K)
inici-1
endi<-ngr[1]
for (i in 1:K) {sumk[1,i]<-sum((muki[ini:end]^2)/(muki[ini:end]*(1-
  (1+sigma2re)*muki[ini:end])))
  inici<end
  endi<-ngr[i+1]+end
}

#Finally, we define the elements of the inverse of V.

vkki<-list(matrix(,nrow=ngr[1],ncol=1))
for (i in 1:K) {
  Vki[[i]]<-matrix(0,nrow=ngr[1,i],ncol=ngr[1,i])
  for (i in 1:ngr[1,i]) {
    yy=1-(1+sigma2re)*muki[i]
    Vki[[i]][i,i]<-1/yy
    Vki[[i]][i,(i+1:(nrow-Vki[[i]][i,i]))]<-(sigma2re*(1/yy)*(1-
      (1+sigma2re)*muki[ini:end])**2)/(muki[ini:end]*(1-
      (1+sigma2re)*muki[ini:end]))
  }
}

chunk

#NAIVE AND SANDWICH ESTIMATORS for IRM#.

#Vectors of individual responses for each group. For ngr[4] is NA but we don't use it.

Ykilist<-list(matrix(,nrow=4,ncol=1))

#Vectors of mean for each group.
mukilist<-list(matrix(,nrow=4,ncol=1))
inici-1
endi<-ngr[1]
for (i in 1:K) {Ykilist[i]<-list(matrix(Dataprove[,5+j],nrow=ngr[i],ncol=1))
  mukilist[i]<-list(matrix(muki[ini:end],nrow=ngr[i],ncol=1))
  inici<end
  endi<-ngr[i+1]+end
}

#Vector diference response and mean.

Ykiminusmuki<-list(matrix(,nrow=4,ncol=1))
for (i in 1:K) {Ykiminusmuki[[i]]<-Ykilist[[i]]-mukilist[[i]]}

#Matrix Dk for each group.

Dkilist<-list(matrix(,nrow=4,ncol=1))
inici-1
endi<-ngr[1]
for (i in 1:K) {Dkilist[i]<-matrix(,nrow=ngr[i],ncol=1+p+1)
  inici<end
  endi<-ngr[i+1]+end
}

ElementKM<-list(matrix(,nrow=4,ncol=1))
Mlist<-list(matrix(0,nrow=(p+1),ncol=(p+1)))
for (i in 1:K) {
  ElementKM[[i]] <-
  t(Dkilist[[i]])%*%Vki[[i]]%*%(Ykiminusmuki[[i]])%*%Vki[[i]]%*%t(Dkilist[[i]])
  Mlist[[1]] <- Mlist[[1]] + ElementKM[[i]]
}

M <- matrix(nrow=(p+1),ncol=(p+1))
for (i in 1:(p+1)) {
  for (j in 1:(p+1)) {M[i,j] <- Mlist[[1]][i,j]}
}

ElementKHs <- list(matrix(,nrow=K,ncol=1))
Hslist <- list(matrix(0,nrow=(p+1),ncol=(p+1)))
for (i in 1:K) {
  ElementKHs[[i]] <- -1*t(Dkilist[[i]])%*%Vki[[i]]%*%Dkilist[[i]]
  Hslist[[1]] <- Hslist[[1]] + ElementKHs[[i]]
}

Hs <- matrix(nrow=(p+1),ncol=(p+1))
for (i in 1:(p+1)) {
  for (j in 1:(p+1)) {Hs[i,j] <- Hslist[[1]][i,j]}
}

naive = ginv(-Hs)
robust = naive%*%M%*%naive

list(naive=naive, robust=robust)
}

#############################################################
##FUNCTION AGGREGATED RANDOM EFFECTS MODEL (fagremcoverage)##
##FOR COMPUTE NAIVE AND SANDWICH ESTIMATOR ####################
#############################################################

fagremcoverage <- function(betain, data) {
  betanew <- as.vector(betain[1,], mode="numeric")
  Dataprove <- data

  # K is the number of groups. We suppose that groups are ordered and they have all.
  # the correlatives numbers. For example:1,2,3 and not 1,3 (There are no number 2).
  # N is the number of observations and p is the number of covariates.
  N <- dim(Dataprove)[1]
  K <- Dataprove[N,2]
  p <- dim(Dataprove)[2]-5

  ngr <- matrix(nrow=1,ncol=K)
  for (i in 1:K) ngr[1,i] <- dim(subset(Dataprove,Dataprove[2]==i))[1]

  gamma0 <- betanew[1]
  beta <- betanew[-1]

  # Individual mean (muki).
  muki <- matrix(nrow=N,ncol=1)
  muki[,1] <- exp(gamma0+as.vector(as.matrix(Dataprove[,6:(5+p)])%*%beta))

  # Individual matrix D.
  Dki <- matrix(nrow=N,ncol=p+1)
  Dki[,1] <- muki[,1]
  for (j in 1:p){
    Dki[,j+1] <- as.numeric(Dataprove[,5+j])*muki[,1]
  }

  # Variance for the ARM.
  muki2 <- matrix(nrow=N,ncol=1)
  muki2[,1] <- muki[,1]^2

  # Outcome for the ARM as defined in Sheppard and Prentice (Biometrics,1995).
  Y <- matrix(nrow=1,ncol=K)
  Y <- matrix(nrow=1,ncol=K)
  phik <- matrix(nrow=K,ncol=1)
  matrix D for the ARM.
  Dk <- matrix(nrow=K,ncol=p+1)
  # First, we compute sigma square.
  sigma2amk <- matrix(nrow=K,ncol=1)
  ini <- 1
  end <- ngr[1]
  for (i in 1:K) {
    Y[1,i] <- ((Dataprove[ini,4])/(Dataprove[ini,5]))
    muki[1,i] <- sum(muki[ini:end])/ngr[1]
    t(Phik[1,i]) <- sum(muki2[ini:end])/ngr[1]
    for (j in 1:(p+1)) {Dki[j,i] <- sum(Dki[ini:end,j])/ngr[1]}
    sigma2amk[i,] <- max((((Y[,1]-muki[,1])^2-(muki[,1]^-phik[,1]/(Dataprove[ini,5])))/(muki[,1]^2-phik[,1])^2),-100)}
ini<-end+1
dend<-ngr[i+1]+end}
sigma2am<-sum(sigma2amk[1:K])/K
#Finally, we define the variance.
Vk<-matrix(,nrow=1,ncol=K)
ini<-1
dend<-ngr[1]
for (i in 1:K) {Vk[i,]<-sigma2am*((muk[i]^2)-(phik[i,]/(Dataprove[ini,5])))+(muk[i]-
phik[i,])*1/(Dataprove[ini,5]))
ini<-1
dend<-ngr[1]}
#NAIVE AND SANDWICH ESTIMATORS for ARM#.
Dkt<-t(Dk)
ElementKMa<-list(matrix(,nrow=K,ncol=1))
Malist<-list(matrix(0,nrow=(p+1),ncol=(p+1)))
for (i in 1:K) {ElementKMa[[i]]<-
matrix(Dkt[,i,](p+1),1,(p+1))*((1/Vk[i])*(Y[i]-muk[i]))^2
Malist[[1]]<Malist[[1]]+ElementKMa[[i]]
}
Ma<-matrix(,nrow=(p+1),ncol=(p+1))
for (i in 1:(p+1)){
for (j in 1:(p+1)) {Ma[i,j]<-Malist[[1]][i,j]}}
ElementKAHs<-list(matrix(,nrow=K,ncol=1))
Hsalist<-list(matrix(0,nrow=(p+1),ncol=(p+1)))
for (i in 1:K) {ElementKAHs[[i]]<-
1*matrix(Dkt[,i,],nrow=(p+1),ncol=1)%%(1/Vk[i])*%Dk[i,]
Hsalist[[1]]<Hsalist[[1]]+ElementKAHs[[i]]
}
Hsa<-matrix(,nrow=(p+1),ncol=(p+1))
for (i in 1:(p+1)){
for (j in 1:(p+1)) {Hsa[i,j]<Hsalist[[1]][i,j]}}
naive=ginv(-Hsa)
robust=naive%*%Ma%*%naive
list(naive=naive, robust=robust)

fpbmcoverage<-function(betain, data) {
betanew<-as.vector(betain[1,],mode="numeric")
Dataprove=data
#K is the number of groups. We suppose that groups are ordered and they have all
#the correlatives numbers. For example:1,2,3 and not 1,3 (There are no number 2).
#N is the number of observations and p is the number of covariates.
N<-dim(Dataprove)[1]
K<-Dataprove[N,2]
p<-dim(Dataprove)[2]-5
ngr<-matrix(,nrow=1,ncol=K)
for (i in 1:K) ngr[i,]<dim(subset(Dataprove,Dataprove[2]==i))[1]
gamma0<-betanew[1]
beta<-betanew[-1]
#Individual outcome.
Yki<-matrix(,nrow=N,ncol=1)
Yki[,1]<Dataprove[,3]
#Individual mean.
muki<-matrix(,nrow=N,ncol=1)
muki[,1]<exp(gamma0+as.vector(as.matrix(Dataprove[,6:(5+p)]))%*%beta)
#Individual matrix D.
Dki<-matrix(,nrow=N,ncol=p+1)
Dki[,1]<muki[,1]
for (j in 1:p) {
Dki[,j+1]<as.numeric(Dataprove[,5+j])*muki[,1]}
#Inverse variance-covariance matrix individual part.
#First, we compute sigma square for the individual part.

```r
muki2 <- matrix(.nrow=N, ncol=1)
muki2[,1] <- muki[,1]^2
Yaver <- matrix(.nrow=1, ncol=K)
muk <- matrix(.nrow=1, ncol=K)
phik <- matrix(.nrow=K, ncol=1)
sigma2rek <- matrix(.nrow=K, ncol=1)
ini <- 1
end <- ngr[1]
for (i in 1:K) {
  Yaver[1,i] <- sum(Yki[ini:end])/ngr[i]
muk[1,i] <- sum(muki[ini:end])/ngr[i]
phik[i,1] <- sum(muki2[ini:end])/ngr[i]
sigma2rek[i,1] <- max(Yaver[1,i]^2-2*ngr[i]*muk[1,i]-
  1+2*((t(muki[ini:end,1])%*%Yki[ini:end,1])/ngr[i])/(ngr[i]-(muk[1,i]^2)-phik[i,1])+1,-
  100)
  ini <- ini+1
  end <- ngr[i+1]+end

  sigma2re <- sum(sigma2rek[1:K])/K
}
#We compute the expression for one part (transpose(muk)*Inverse(Deltak)*muk).

sumk <- matrix(.nrow=1, ncol=K)
ini <- 1
end <- ngr[1]
for (i in 1:K) {sumk[1,i] <- sum((muki[ini:end]^2)/(muki[ini:end]*(1-(1+sigma2re)*muki[ini:end])))
  ini <- ini+1
  end <- ngr[i+1]+end}
#Finally we define the elements of the inverse of V.

Vki <- list(matrix(.nrow=1, ncol=1))
for (j in 1:K) {
  Vki[[j]] <- matrix(0, nrow=ngr[1,j], ncol=ngr[1,j])
  for (i in 1:ngr[1,j]) {
    yy <- 1-(1+sigma2re)*muki[i]
    Vki[[j]][i,] <- -(sigma2re*(1/yy)*(1/(1-
      (1+sigma2re)*muki[i])))*(1/(1+sigma2re*sumk[1,j]))-
      (sigma2re*(1/yy)^2)*(1/(1+sigma2re*sumk[1,j]))
  }
  Vki[[j]] = Vki[[j]] + t(Vki[[j]]) - diag(diag(Vki[[j]]))
}
#Vectors of individual responses for each group. For ngr[4] is NA but we don't use it.

Ykilist <- list(matrix(.nrow=1, ncol=1))
#Vectors of mean for each group.

mukilist <- list(matrix(.nrow=1, ncol=1))
ini <- 1
end <- ngr[1]
for (i in 1:K) {Ykilist[i] <- list(matrix(Dataprove[ini:end,3], nrow=ngr[i], ncol=1))
mukilist[i] <- list(matrix(muki[ini:end], nrow=ngr[i], ncol=1))
ini <- ini+1
end <- ngr[i+1]+end}
#Vector difference response and mean.

Ykiminusmuki <- list(matrix(0, nrow=ngr[1,1], ncol=ngr[1,1]))

#Matrix Dk for each group.

Dkilist <- list(matrix(.nrow=ngr[1,1], ncol=p+1))
ini <- 1
end <- ngr[1]
for (j in 1:K) {Dkilist[[j]] <- matrix(0, nrow=ngr[1,1], ncol=p+1)
  ini <- ini+1
  end <- ngr[j+1]+end}
#Outcome for the aggregated data model with combined analytical and aggregated models.

Ybar <- matrix(.nrow=1, ncol=K)

#Matrix D for the aggregated part.

Dk <- matrix(.nrow=K, ncol=p+1)
ini <- 1
end <- ngr[1]
```
for (i in 1:K) {Ybar[i]<-((Dataprove[ini,4]-sum(Yki[ini:end]))/(Dataprove[ini,5]-ngr[i]))}
for (j in 1:(p+1)) {Dk[i,j]<-sum(Dki[ini:end,j])/ngr[i]}
in<ini+1
end<ng[i+1]+end

Dkt<-t(Dk)

#Sigma square aggregated part.
sigma2p<matrix(nrow=K,ncol=1)
in<1
end<ng[1]
for (i in 1:K) {sigma2p[i]<max((((Ybar[i]-muk[i])^2*(muk[i]-phik[i]/(Dataprove[ini,5]-ngr[i]))+(muk[i]-phik[i]/(Dataprove[ini,5]-ngr[i]))))/((Ybar[i]-muk[i])^2-(muk[i]-phik[i]/(Dataprove[ini,5]-ngr[i]))),0)}
in<ini+1
end<ng[i+1]+end

sigma2pb<-sum(sigma2p[1:K])/K

Vkbar<-matrix(nrow=1,ncol=K)
Mpb=matrix(0,nrow=(p+1),ncol=(p+1))
in<1
for (i in 1:K) {
Vkbar[i]<-sigma2pb*((muk[i]^2)-(phik[i]/(Dataprove[ini,5]-ngr[i])))+(muk[i]-phik[i]/(Dataprove[ini,5]-ngr[i]))
}
junkmat=diag(c(rep(0,ng[i]),1/Vkbar[i]))
junkmat[1:ng[i],1:ng[i]]=Vki[i]
sub=t(rbind(Dkilist[i],Dk[i,]))%*%junkmat%*%rbind(Ykiminusmuki[i],Ybar[i]-muk[i])
Mpb=Mpb+sub%*%t(sub)
in<ini+1
end<ng[i+1]+end

ElementKHs<-list(matrix(nrow=K,ncol=1))
Hslist<-list(matrix(0,nrow=(p+1),ncol=(p+1)))
for (i in 1:K) {
ElementKHs[i]<-1*t(Dkilist[i])%*%Vki[i]%*%Dkilist[i]
}

Hs<-matrix(nrow=(p+1),ncol=(p+1))
for (i in 1:K) {
for (j in 1:K) {Hs[i,j]<-Hslist[i,j]}}

Hsar<-matrix(nrow=(p+1),ncol=(p+1))
for (i in 1:K) {
for (j in 1:K) {Hsar[i,j]<-Hsarlist[i,j]}}

Hspb<-Hs+Hsar

naive=ginv(-Hspb)
robust=naive%*%Mpb%*%naive

list(naive=naive, robust=robust)


###########################################################
#CONFIDENCE INTERVAL NAIVE AND SANDWICH FOR THE IRM####
##(function farem2coverage, ARM (function fagrem2coverage)##
##AND PBEE (function fpbm2coverage)#############
###########################################################

farem2coverage<-function(data,datavar,beta1=.2){
result=rep(NA,21)
if (datavar[5]<51) {
  q<-faremcoverage(datavar[1:3],data)
  naive=q$naive
  robust=q$robust
}
if (naive[2,2]<0){
  result[2:3]=rep(-1,2)
  result[16]=-1}
if (robust[2,2]<0){
  result[2:3]=rep(-1,2)
  result[16]=-1}
result[4:5]=rep(-1,2)
result[17]=-1

if (naive[2,2]>>=0){
  ll=(datavar[2]-1.96*sqrt(naive[2,2]))
  ul=(datavar[2]+1.96*sqrt(naive[2,2]))
  result[2:3]=c(ll,ul)
  result[16]=ifelse(beta1>=ll & beta1<=ul, 1, 0)
}

if (robust[2,2]>>=0){
  ll=(datavar[2]-1.96*sqrt(robust[2,2]))
  ul=(datavar[2]+1.96*sqrt(robust[2,2]))
  result[4:5]=c(ll,ul)
  result[17]=ifelse(beta1>=ll & beta1<=ul, 1, 0)
}
elseresult[c(2:5,16:17)]=rep(-1,6)
}

unlist(result)
fagrem2coverage<-function(data,datavar,beta=.2){
  result=rep(NA,21)
  if (datavar[10]<51) {
    d<-fagremcoverage(datavar[6:8],data)
    naive=d$naive
    robust=d$robust
    if (naive[2,2]<0){
      result[7:8]=rep(-1,2)
      result[18]=-1
    }
    if (robust[2,2]<0){
      result[9:10]=rep(-1,2)
      result[19]=-1
    }
    if (naive[2,2]>>=0){
      ll=(datavar[7]-1.96*sqrt(naive[2,2]))
      ul=(datavar[7]+1.96*sqrt(naive[2,2]))
      result[7:8]=c(ll,ul)
      result[18]=ifelse(beta1>=ll & beta1<=ul, 1, 0)
    }
    if (robust[2,2]>>=0){
      ll=(datavar[7]-1.96*sqrt(robust[2,2]))
      ul=(datavar[7]+1.96*sqrt(robust[2,2]))
      result[9:10]=c(ll,ul)
      result[19]=ifelse(beta1>=ll & beta1<=ul, 1, 0)
    }
  }else {
    result[c(7:10,18:19)]=rep(-1,6)
  }
  unlist(result)
}

fpbm2coverage<-function(data,datavar,beta=.2){
  result=rep(NA,21)
  if (datavar[16]<51) {
    pb<-fpbmcoverage(datavar[11:13],data)
    naive=pb$naive
    robust=pb$robust
    if (naive[2,2]<0){
      result[12:13]=rep(-1,2)
      result[20]=-1
    }
    if (robust[2,2]<0){
      result[14:15]=rep(-1,2)
      result[21]=-1
    }
    if (naive[2,2]>>=0){
      ll=(datavar[12]-1.96*sqrt(naive[2,2]))
      ul=(datavar[12]+1.96*sqrt(naive[2,2]))
      result[12:13]=c(ll,ul)
      result[20]=ifelse(beta1>=ll & beta1<=ul, 1, 0)
    }
    if (robust[2,2]>>=0){
      ll=(datavar[12]-1.96*sqrt(robust[2,2]))
      ul=(datavar[12]+1.96*sqrt(robust[2,2]))
      result[14:15]=c(ll,ul)
      result[21]=ifelse(beta1>=ll & beta1<=ul, 1, 0)
    }
  }else {
    result[c(12:15,20:21)]=rep(-1,6)
  }
  unlist(result)
ul=(datavar[12]+1.96*sqrt(naive[2,2]))
result[12:13]=c(ll,ul)
result[20]=ifelse(beta1>=ll & beta1<=ul, 1, 0)
}
if (robust[2,2]>=0){
  ll=(datavar[12]-1.96*sqrt(robust[2,2]))
  ul=(datavar[12]+1.96*sqrt(robust[2,2]))
  result[14:15]=c(ll,ul)
  result[21]=ifelse(beta1>=ll & beta1<=ul, 1, 0)
}
else {
  result[c(12:15,20:21)]=rep(-1,6)
}
unlist(result)

**************************************************************************.
#FUNCTION FOR SIMULATE THE DATA (fgenerate2)##.
#IN THE SIMPLE CONFOUNDSING CASE (SCC)     ##.
**************************************************************************.
fgenerate2<-function(group,populationsize,samplesize,variance){
  #group=number of groups, populationsize= population size.
  #samplesize= sample size, variance=within group variance.
  K<-group
  nk<-populationsize
  mk<-samplesize
  varwithin<-variance
  #Covariate X1ki & X2ki. They are correlated 0.3 at the community.
  #and individual levels (see Prentice & Sheppard).
  zk=mvrnorm(K,c(0,0),matrix(c(1,.3,.3,1),2,2))
  cov=0.3*sqrt(varwithin)+1
  covm=matrix(c(varwithin,cov,cov,1),2,2)
  X1ki=matrix(0,nk,K)
  X2ki=matrix(0,nk,K)
  for(i in 1:K){
    znk=mvrnorm(nk,zk[i,],covm)
    X1ki[,i]=znk[,1]
    X2ki[,i]=znk[,2]
  }
  #Country specific frailties were generated as independent.
  #realized values from a gamma distribution with mean 1.
  #and the variance is shape*scale. The mean of a gamma is shape*scale.
  #and the variance is shape^2*scale^2.
  meanhk<-1
  varhk<-0.05
  shape<-(meanhk^2)/(varhk)
  scale<-(varhk)/(meanhk)
  hk<-rgamma(K,shape=shape,scale=scale)
  hk=t(matrix(rep(hk,nk),nrow=K,ncol=nk))
  #The disease events, yki, were generated by determining.
  #whether a uniform random variable was less than.
  #hk*exp(gamma0+beta1*X1ki+beta2*X2ki).
  gamma0<-3
  beta1<-0.2
  beta2<-0.2
  yki<-matrix(,nrow=nk,ncol=K)
  unif<-matrix(runif(nk*K,0,1),nrow=nk,ncol=K)
  yki=ifelse(unif*exp(gamma0+beta1*X1ki+beta2*X2ki),1,0)
  dataolist<-list(matrix(,nrow=K,ncol=1))
  sampledatalist<-list(matrix(,nrow=K,ncol=1))
}
ini<-1
end<-mk

data<-matrix(nrow=mk*K,ncol=5)
for (i in 1:K){

datalist[[i]]<-cbind(matrix(c(1:nk),nrow=nk,ncol=1),matrix(yki[,i],nrow=nk,ncol=1),matrix(X1ki[,i],nrow=nk,ncol=1),matrix(X2ki[,i],nrow=nk,ncol=1),matrix(c(i),nrow=nk,ncol=1))
sampledatalist[[i]]<-datalist[[i]][as.matrix(sample(datalist[[i]][,1],1),mk])
data[ini:end,]<-sampledatalist[[i]]

ini<-end+1
end<-mk*(i+1)
}

O<-matrix(apply(yki,2,sum),nrow=K,ncol=1)
ini<-1
end<-mk

datapop<-matrix(nrow=mk*K,ncol=1)
for (i in 1:K) {datapop[ini:end,1]<-O[i,]
ini<-end+1
end<-mk*(i+1)
}
datadatapop<-cbind(data,datapop,c(nk))
datafin<-data.frame(id=matrix(datadatapop[,1]),group=matrix(datadatapop[,5]),YIND=matrix(datadatapop[,2]),O=matrix(datadatapop[,6]),n=matrix(datadatapop[,7]),X1ki=matrix(datadatapop[,3]),X2ki=matrix(datadatapop[,4]))


##########################################.
#100 groups-100 sample size in each group#.
##########################################.
#The next text files are the results files from the simulation runs.
#To obtain parameter estimates in each variation ratio for the 100-100 case.
#For example, dataA1var1.txt is from finalresultA100100A.25$result.
dataB1var1<-read.table("dataB1var1.txt",header=T) #Variance 0.25.
dataB1var2<-read.table("dataB1var2.txt",header=T) #Variance 0.5.
dataB1var3<-read.table("dataB1var3.txt",header=T) #Variance 1.
dataB1var4<-read.table("dataB1var4.txt",header=T) #Variance 2.
dataB1var5<-read.table("dataB1var5.txt",header=T) #Variance 4.
dataB1var6<-read.table("dataB1var6.txt",header=T) #Variance 8.
dataB1var7<-read.table("dataB1var7.txt",header=T) #Variance 16.

coverage100100B.25=fsimulationBcoverage(123,1000,.25,100,100,dataB1var1)
save(list=c("coverage100100B.25",".Random.seed"),file="100100coverageB025.RData")
savedseed=.Random.seed

coverage100100B.5=fsimulationBcoverage(savedseed,1000,.5,100,100,dataB1var2)
save(list=c("coverage100100B.5",".Random.seed"),file="100100coverageB05.RData")
savedseed=.Random.seed

coverage100100B1=fsimulationBcoverage(savedseed,1000,1,100,100,dataB1var3)
save(list=c("coverage100100B1",".Random.seed"),file="100100coverageB1.RData")
savedseed=.Random.seed

coverage100100B2=fsimulationBcoverage(savedseed,1000,2,100,100,dataB1var4)
save(list=c("coverage100100B2",".Random.seed"),file="100100coverageB2.RData")
savedseed=.Random.seed

coverage100100B3=fsimulationBcoverage(savedseed,1000,3,100,100,dataB1var5)
save(list=c("coverage100100B3",".Random.seed"),file="100100coverageB3.RData")
savedseed=.Random.seed

coverage100100B4=fsimulationBcoverage(savedseed,1000,4,100,100,dataB1var6)
save(list=c("coverage100100B4",".Random.seed"),file="100100coverageB4.RData")
savedseed=.Random.seed

coverage100100B5=fsimulationBcoverage(savedseed,1000,5,100,100,dataB1var7)
save(list=c("coverage100100B5",".Random.seed"),file="100100coverageB5.RData")

coverage100100B4=fsimulationBcoverage(savedseed,1000,4,100,100,dataB1var5)
save(list=c("coverage100100B4" ,".Random.seed"),file="100100coverageB4.RData")
savedseed=.Random.seed
coverage100100B8=fsimulationBcoverage(savedseed,1000,8,100,100,dataB1var6)
save(list=c("coverage100100B8" ,".Random.seed"),file="100100coverageB8.RData")
savedseed=.Random.seed
coverage100100B16=fsimulationBcoverage(savedseed,1000,16,100,100,dataB1var7)
save(list=c("coverage100100B16" ,".Random.seed"),file="100100coverageB16.RData")

#50 groups-100 sample size in each group#

#The next text files are the results files from the simulation runs.
#to obtain parameter estimates in each variation ratio for the 50-100 case.

dataB2var1<-read.table("dataB2var1.txt",header=T) #Variance 0.25.
dataB2var2<-read.table("dataB2var2.txt",header=T) #Variance 0.5.
dataB2var5<-read.table("dataB2var5.txt",header=T) #Variance 4.
dataB2var7<-read.table("dataB2var7.txt",header=T) #Variance 16.

coverage50100B.25=fsimulationBcoverage(123,1000,.25,50,100,dataB2var1)
save(list=c("coverage50100B.25" ,".Random.seed"),file="50100coverageB025.RData")
savedseed=.Random.seed
coverage50100B.5=fsimulationBcoverage(savedseed,1000,.5,50,100,dataB2var2)
save(list=c("coverage50100B.5" ,".Random.seed"),file="50100coverageB05.RData")
savedseed=.Random.seed
coverage50100B1=fsimulationBcoverage(savedseed,1000,1,50,100,dataB2var3)
save(list=c("coverage50100B1" ,".Random.seed"),file="50100coverageB1.RData")
savedseed=.Random.seed
coverage50100B2=fsimulationBcoverage(savedseed,1000,2,50,100,dataB2var4)
save(list=c("coverage50100B2" ,".Random.seed"),file="50100coverageB2.RData")
savedseed=.Random.seed
coverage50100B4=fsimulationBcoverage(savedseed,1000,4,50,100,dataB2var5)
save(list=c("coverage50100B4" ,".Random.seed"),file="50100coverageB4.RData")
savedseed=.Random.seed
coverage50100B8=fsimulationBcoverage(savedseed,1000,8,50,100,dataB2var6)
save(list=c("coverage50100B8" ,".Random.seed"),file="50100coverageB8.RData")
savedseed=.Random.seed
coverage50100B16=fsimulationBcoverage(savedseed,1000,16,50,100,dataB2var7)
save(list=c("coverage50100B16" ,".Random.seed"),file="50100coverageB16.RData")

#100 groups-50 sample size in each group#

#The next text files are the results files from the simulation runs.
#to obtain parameter estimates in each variation ratio for the 50-100 case.

dataB3var1<-read.table("dataB3var1.txt",header=T) #Variance 0.25.
dataB3var2<-read.table("dataB3var2.txt",header=T) #Variance 0.5.
dataB3var3<-read.table("dataB3var3.txt",header=T) #Variance 1.
dataB3var4<-read.table("dataB3var4.txt",header=T) #Variance 2.
dataB3var5<-read.table("dataB3var5.txt",header=T) #Variance 4.
dataB3var6<-read.table("dataB3var6.txt",header=T) #Variance 8.
dataB3var7<-read.table("dataB3var7.txt",header=T) #Variance 16.

coverage10050B.25=fsimulationBcoverage(123,1000,.25,25,50,100,dataB3var1)
save(list=c("coverage10050B.25" ,".Random.seed"),file="10050coverageB025.RData")
savedseed=.Random.seed
coverage10050B.5=fsimulationBcoverage(savedseed,1000,.5,25,50,100,dataB3var2)
save(list=c("coverage10050B.5" ,".Random.seed"),file="10050coverageB05.RData")
savedseed=.Random.seed
coverage10050B1=fsimulationBcoverage(savedseed,1000,1,25,50,100,dataB3var3)
save(list=c("coverage10050B1" ,".Random.seed"),file="10050coverageB1.RData")
savedseed=.Random.seed
coverage10050B2=fsimulationBcoverage(savedseed,1000,2,25,50,100,dataB3var4)
save(list=c("coverage10050B2" ,".Random.seed"),file="10050coverageB2.RData")
savedseed=.Random.seed
coverage10050B4=fsimulationBcoverage(savedseed,1000,4,25,50,100,dataB3var5)
save(list=c("coverage10050B4", ".Random.seed"),file="10050coverageB4.RData")
savedseed=.Random.seed
coverage10050B8=fsimulationBcoverage(savedseed,1000,8,100,50,dataB3var6)
save(list=c("coverage10050B8", ".Random.seed"),file="10050coverageB8.RData")
savedseed=.Random.seed
coverage10050B16=fsimulationBcoverage(savedseed,1000,16,100,50,dataB3var7)
save(list=c("coverage10050B16", ".Random.seed"),file="10050coverageB16.RData")

#50 groups-50 sample size in each group#

#The next text files are the results files from the simulation runs.
#to obtain parameter estimates in each variation ratio for the 50–100 case.

dataB4var1<-read.table("dataB4var1.txt",header=T) #Variance 0.25.
dataB4var2<-read.table("dataB4var2.txt",header=T) #Variance 0.5.
dataB4var3<-read.table("dataB4var3.txt",header=T) #Variance 1.
dataB4var4<-read.table("dataB4var4.txt",header=T) #Variance 2.
dataB4var5<-read.table("dataB4var5.txt",header=T) #Variance 4.
dataB4var6<-read.table("dataB4var6.txt",header=T) #Variance 8.
dataB4var7<-read.table("dataB4var7.txt",header=T) #Variance 16.

coverage5050B.25=fsimulationBcoverage(123,1000,.25,50,50,dataB4var1)
save(list=c("coverage5050B.25", ".Random.seed"),file="5050coverageB025.RData")
savedseed=.Random.seed
coverage5050B.5=fsimulationBcoverage(savedseed,1000,.5,50,50,dataB4var2)
save(list=c("coverage5050B.5", ".Random.seed"),file="5050coverageB05.RData")
savedseed=.Random.seed
coverage5050B1=fsimulationBcoverage(savedseed,1000,1,50,50,dataB4var3)
save(list=c("coverage5050B1", ".Random.seed"),file="5050coverageB1.RData")
savedseed=.Random.seed
coverage5050B2=fsimulationBcoverage(savedseed,1000,2,50,50,dataB4var4)
save(list=c("coverage5050B2", ".Random.seed"),file="5050coverageB2.RData")
savedseed=.Random.seed
coverage5050B4=fsimulationBcoverage(savedseed,1000,4,50,50,dataB4var5)
save(list=c("coverage5050B4", ".Random.seed"),file="5050coverageB4.RData")
savedseed=.Random.seed
coverage5050B8=fsimulationBcoverage(savedseed,1000,8,50,50,dataB4var6)
save(list=c("coverage5050B8", ".Random.seed"),file="5050coverageB8.RData")
savedseed=.Random.seed
coverage5050B16=fsimulationBcoverage(savedseed,1000,16,50,50,dataB4var7)
save(list=c("coverage5050B16", ".Random.seed"),file="5050coverageB16.RData")

###COVERAGE INTERVAL OF THE ESTIMATES ##.

#covinter<-function(result,Niter){
#result is the file with the estimate parameter b1 and the confidence
#interval with the naive estimator and sandwich in the IRM, ARM & PBEE.

covinter<-function(result,Niter){
  resultaremsubnaive=matrix(subset(resultarem2,resultarem2[,1]>=0),ncol=3)
  resultagremsubnaive=matrix(subset(resultagrem2,resultagrem2[,1]>=0),ncol=3)
  resultpbmsubnaive=matrix(subset(resultpbm2,resultpbm2[,1]>=0),ncol=3)
  resultaremsubsandwich=matrix(subset(resultarem2,resultarem2[,2]>=0),ncol=3)
  resultagremsubsandwich=matrix(subset(resultagrem2,resultagrem2[,2]>=0),ncol=3)
  resultpbmsubsandwich=matrix(subset(resultpbm2,resultpbm2[,2]>=0),ncol=3)

  sumaremnaive=sum(resultaremsubnaive[,1])
  naremnaive=sum(resultaremsubnaive[,3])
  sumagremsubnaive=sum(resultagremsubnaive[,1])
  nagremsubnaive=sum(resultagremsubnaive[,3])
  sumpbmsubnaive=sum(resultpbmsubnaive[,1])
  npbmsubnaive=sum(resultpbmsubnaive[,3])
sumaremsandwich=sum(resultaremsubsandwich[,2])
naremsandwich=sum(resultaremsubsandwich[,3])
sumagremsandwich=sum(resultagremsubsandwich[,2])
nagremsandwich=sum(resultagremsubsandwich[,3])
sumpbmsandwich=sum(resultpbmsubsandwich[,2])
npbmsandwich=sum(resultpbmsubsandwich[,3])

#coverage interval for IRM.
aremnaive=sumaremnaive/naremnaive
aremsandwich=sumaremsandwich/naremsandwich

#coverage interval for ARM.
agremnaive=sumagremnaive/nagremnaive
agremsandwich=sumagremsandwich/nagremsandwich

#coverage interval for PBEE.
pbmnaive=sumpbmnaive/npbmnaive
pbmsandwich=sumpbmsandwich/npbmsandwich

cat("Coverage interval naive
(AREM,AGREM,PBM)",
aremnaive,
agremnaive,
pbmnaive,
"
",
aremsandwich,
agremsandwich,
pbmsandwich,
"
")

A.3.6 Bias and mean square error program.

###BIAS AND MEAN SQUARE ERROR OF ESTIMATES ###.

#The function fbiasmse computes the bias and mean square error for the results.
#of the simulations (The simulation file has the next structure: first the:
#parameters of the IRM: gamma0, beta1, beta2, variance, number of iterations. The.
#next columns are the parameters for the ARM: gamma0, beta1, beta2, variance,.
#number of iterations. Finally, the parameters for the PBEE: gamma0, beta1, beta2,.
#variance individual, variance aggregated, number of iterations.

fbiasmse=function(result,g0,b1,b2,aremvar,agremvar,pbavar,pbagvar){
  #result is the file with the runs of the simulation, g0 is the true value of.
  #the true value of b1, b2 is the true value of b2, aremvar is.
  #the true value of the variance of the IRM, agremvar is the true value of the.
  #variance of the ARM, pbavar is the true value of the variance of the individual.
# part of the PBEE, pbagvar  is the true value of the variance of the aggregated.
# part of the PBEE.

resultarem<-matrix(result[,1:5],ncol=5)
resultagrem<-matrix(result[,6:10],ncol=5)
resultpbm<-matrix(result[,11:16],ncol=6)
resultaremsub<-matrix(subset(resultarem,resultarem[,5]<51),ncol=5)
resultagremsub<-matrix(subset(resultagrem,resultagrem[,5]<51),ncol=5)
resultpbmsub<-matrix(subset(resultpbm,resultpbm[,6]<51),ncol=6)

meanarem<-apply(resultarem,2,mean)
meanagrem<-apply(resultagrem,2,mean)
meannpbm<-apply(resultpbm,2,mean)

vararem<-apply(resultarem,2,var)
varagrem<-apply(resultagrem,2,var)
varpbm<-apply(resultpbm,2,var)

#bias IRM.
arembiasgamma0=meanarem[1]-g0
arembiasbeta1=meanarem[2]-b1
arembiasbeta2=meanarem[3]-b2
arembiasvariance=meanarem[4]-aremvar

#bias ARM.
agrembiasgamma0=meanagrem[1]-g0
agrembiasbeta1=meanagrem[2]-b1
agrembiasbeta2=meanagrem[3]-b2
agrembiasvariance=meanagrem[4]-agremvar

#bias PBEE.
pbmbiasgamma0=meanpbm[1]-g0
pbmbiasbeta1=meanpbm[2]-b1
pbmbiasbeta2=meanpbm[3]-b2
pbmbiasvarianceanalytical=meanpbm[4]-pbavar
pbmbiasvarianceaggregated=meanpbm[5]-pbagvar

#MSE IRM.
aremmssegamma0=apply((matrix(resultarem[,1])-g0)^2,2,mean)
amremmsbeta1=apply((matrix(resultarem[,2])-b1)^2,2,mean)
amremmsbeta2=apply((matrix(resultarem[,3])-b2)^2,2,mean)
amremmsvariance=apply((matrix(resultarem[,4])-aremvar)^2,2,mean)

#MSE ARM.
agremmssegamma0=apply((matrix(resultagrem[,1])-g0)^2,2,mean)
agremmsbeta1=apply((matrix(resultagrem[,2])-b1)^2,2,mean)
agremmsbeta2=apply((matrix(resultagrem[,3])-b2)^2,2,mean)
agremmsvariance=apply((matrix(resultagrem[,4])-agremvar)^2,2,mean)

#MSE PBEE.
pbmmsegamma0=apply((matrix(resultpbm[,1])-g0)^2,2,mean)
pbmmsebeta1=apply((matrix(resultpbm[,2])-b1)^2,2,mean)
pbmmsebeta2=apply((matrix(resultpbm[,3])-b2)^2,2,mean)
pbmmsevarianceanalytical=apply((matrix(resultpbm[,4])-pbavar)^2,2,mean)
pbmmsevarianceaggregated=apply((matrix(resultpbm[,5])-pbagvar)^2,2,mean)

cat("Bias b1 (IRM, ARM, PBEE)","\n",round((arembiasbeta1/0.2)*100,digit=2),"\n",round((agrembiasbeta1/0.2)*100,digit=2),"\n")
cat("Mse b1 (IRM, ARM, PBEE)","\n",round(aremmsebeta1*100,digit=2),"\n",round(agremmsbeta1*100,digit=2),"\n")

cat("Bias b2 (IRM, ARM, PBEE)","\n",round((arembiasbeta2/0.2)*100,digit=2),"\n",round((agrembiasbeta2/0.2)*100,digit=2),"\n")
cat("Mse b2 (IRM, ARM, PBEE)","\n",round(aremmsebeta2*100,digit=2),"\n",round(agremmsbeta2*100,digit=2),"\n")

#NON-CONF edLING CASE.

#100 groups-100 sample size in each group.
fbiasmse(finalresultA100100A.25$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultA100100A.5$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultA100100A1$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultA100100A2$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultA100100A4$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultA100100A8$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultA100100A16$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)

#100 groups-50 sample size in each group.
fbiasmse(finalresultA10050A.25$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultA10050A$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultA10050A1$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)

232
fbiasmse(finalresultA10050A2$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultA10050A4$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultA10050A8$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultA10050A16$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)

#50 groups-100 sample size in each group.
fbiasmse(finalresultA50100A.25$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultA50100A.5$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultA50100A1$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultA50100A2$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)

fbiasmse(finalresultA50100A4$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultA50100A8$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultA50100A16$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)

#SIMPLE CONFOUNDING CASE.

#100 groups-100 sample size in each group.
fbiasmse(finalresultB100100B.25$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB100100B.5$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB100100B1$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB100100B2$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)

fbiasmse(finalresultB100100B4$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB100100B8$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB100100B16$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)

#100 groups-50 sample size in each group.

fbiasmse(finalresultB10050B.25$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB10050B.5$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB10050B1$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB10050B2$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)

fbiasmse(finalresultB10050B4$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB10050B8$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB10050B16$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)

#50 groups-100 sample size in each group.

fbiasmse(finalresultB50100B.25$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB50100B.5$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB50100B1$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB50100B2$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)

fbiasmse(finalresultB50100B4$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB50100B8$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB50100B16$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)

#50 groups-50 sample size in each group.

fbiasmse(finalresultB5050B.25$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB5050B.5$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB5050B1$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB5050B2$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)

fbiasmse(finalresultB5050B4$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB5050B8$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
fbiasmse(finalresultB5050B16$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)

#EXTENDED CONFOUNDING CASE.

#100 groups-100 sample size in each group.
fbiasmse(finalresultC100100$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)

#100 groups-50 sample size in each group.
fbiasmse(finalresultC10050$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)

#50 groups-100 sample size in each group.
fbiasmse(finalresultC50100$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)

#50 groups-50 sample size in each group.
fbiasmse(finalresultC5050$result,-3,0.2,0.2,0.05,0.05,0.05,0.05)
A.4 Demonstrations.

A.4.1 Calculation of $\hat{Y}_k^A$.

We know

$$\text{Var}(\overline{Y}_k^A) = \text{Var}(E(\overline{Y}_k^A|h_k)) + E(\text{Var}(\overline{Y}_k^A|h_k)) \quad [A4]$$

First, we compute the quantities $E(\overline{Y}_k^A|h_k)$ and $\text{Var}(\overline{Y}_k^A|h_k)$

$$E(\overline{Y}_k^A|h_k) = E\left(\frac{\sum_{i=1}^{n_k-m_k} Y_{ki}}{n_k-m_k} | h_k\right) = \frac{\sum_{i=1}^{n_k-m_k} E(Y_{ki} | h_k)}{n_k-m_k} = h_k \sum_{i=1}^{n_k-m_k} e^{x_{i}^T \alpha}$$

$$\text{Var}(\overline{Y}_k^A|h_k) = \text{Var}\left(\frac{\sum_{i=1}^{n_k-m_k} Y_{ki}}{n_k-m_k} | h_k\right) = \frac{\sum_{i=1}^{n_k-m_k} \text{Var}(Y_{ki} | h_k)}{(n_k-m_k)^2} = \frac{\sum_{i=1}^{n_k-m_k} (h_k e^{x_{i}^T \alpha} - h_k^2 e^{2x_{i}^T \alpha})}{(n_k-m_k)^2}$$

We know that $h_k$ is a random effect with $E[h_k] = 1$ and $\text{Var}[h_k] = \sigma^2$ and

$$\mu^A = E[\overline{Y}_k^A] = E\left[\frac{h_k \sum_{i=1}^{n_k-m_k} e^{x_{i}^T \alpha}}{n_k-m_k} \right] = \frac{E[h_k] \sum_{i=1}^{n_k-m_k} e^{x_{i}^T \alpha}}{n_k-m_k} = \frac{h_k \sum_{i=1}^{n_k-m_k} e^{x_{i}^T \alpha}}{n_k-m_k}$$

so we can compute [A4] as

$$\text{Var}[\overline{Y}_k^A] = \text{Var}\left[\frac{h_k \sum_{i=1}^{n_k-m_k} e^{x_{i}^T \alpha}}{n_k-m_k}\right] + \text{Var}\left[\frac{\sum_{i=1}^{n_k-m_k} \left(h_k e^{x_{i}^T \alpha} - h_k^2 e^{2x_{i}^T \alpha}\right)}{(n_k-m_k)^2}\right]$$

$$= \text{Var}[h_k \mu^A] + \text{Var}\left[\frac{h_k^2 \phi_k}{n_k-m_k}\right] = (\mu^A)^2 \text{Var}[h_k] + \frac{\mu^A E[h_k^2]}{n_k-m_k} - \frac{\phi_k E[h_k^2]}{n_k-m_k}$$

$$= (\mu^A)^2 \sigma^2 + \frac{\mu^A}{n_k-m_k} - \frac{\phi_k (\sigma^2 + 1)}{n_k-m_k} = (\mu^A)^2 \sigma^2 + [\mu^A - \phi_k (\sigma^2 + 1)](n_k-m_k)^{-1}$$

$$= \sigma^2 (\mu^A)^2 - \phi_k (n_k-m_k)^{-1} + (\mu^A - \phi_k)(n_k-m_k)^{-1}$$

where $\phi_k = \sum_{i=1}^{n_k-m_k} e^{2x_{i}^T \alpha} / n_k-m_k$. We consider $\hat{\phi}_k = \epsilon_m \{e^{2x_{i}^T \alpha}\}$ and $\hat{\mu}^A = \epsilon_m \{e^{x_{i}^T \alpha}\}$ so

$$\hat{Y}_k^A = \sigma^2 ((\hat{\mu}^A)^2 - \hat{\phi}_k (n_k-m_k)^{-1}) + (\hat{\mu}^A - \hat{\phi}_k)(n_k-m_k)^{-1}$$

(1) $Y_u$ follow a Bernoulli$(p_u)$ so $p_u = E(Y_u | h_u) = E(Y_u^2 | h_u)$.
A.4.2 Estimation for $\sigma^2$ in the individual part of the PBEE approach.

We know

$$E[(Y_{ki} - \mu_{ki})(Y_{kj} - \mu_{kj})] = \sigma^2 \mu_{ki}\mu_{kj} \quad \text{for } i \neq j \text{ and } k=1,\ldots,K$$

so

$$\sum_{k=1}^{K} \sum_{i=j}^{m_k} (Y_{ki} - \mu_{ki})(Y_{kj} - \mu_{kj}) = \sum_{k=1}^{K} \sum_{i=j}^{m_k} \sigma^2 \mu_{ki}\mu_{kj}$$

$$\sum_{k=1}^{K} \left( \sum_{i=1}^{m_k} \sum_{j=1}^{m_k} (Y_{ki}Y_{kj} - \mu_{ki}Y_{kj} - Y_{ki}\mu_{kj} + \mu_{ki}\mu_{kj}) - \sum_{i=1}^{m_k} (Y_{ki}^2 - 2\mu_{ki}Y_{ki} + \mu_{ki}^2) \right)$$

$$= \sum_{k=1}^{K} \left( \sigma^2 \sum_{i=1}^{m_k} \sum_{j=1}^{m_k} \mu_{ki}\mu_{kj} - \sum_{i=1}^{m_k} \mu_{ki}^2 \right)$$

$$\sum_{k=1}^{K} \left( \sum_{i=1}^{m_k} \sum_{j=1}^{m_k} (Y_{ki}^2 - 2Y_{ki} \mu_k^A m_k^2 + (\mu_k^A)^2 m_k^2 - \sum_{i=1}^{m_k} Y_{ki}^2 + 2 \sum_{i=1}^{m_k} \mu_{ki}Y_{ki} - \sum_{i=1}^{m_k} \mu_{ki}^2 \right)$$

$$= \sum_{k=1}^{K} \sigma^2 \left( (\mu_k^A)^2 m_k^2 - \sum_{i=1}^{m_k} \mu_{ki}^2 \right)$$

where $\mu_k^A = \frac{\sum_{i=1}^{m_k} \mu_{ki}}{m_k} = \epsilon_{m_k} \{ e^{x_i|\alpha} \}$,

$$\sum_{k=1}^{K} \left( \sum_{i=1}^{m_k} \sum_{j=1}^{m_k} (Y_{ki} m_k - 2\mu_k^A m_k - 1) + Y_{ki} + (\mu_k^A)^2 m_k - \frac{\sum_{i=1}^{m_k} Y_{ki}^2}{m_k} + \frac{2 \sum_{i=1}^{m_k} \mu_{ki}Y_{ki}}{m_k} - \frac{\sum_{i=1}^{m_k} \mu_{ki}^2}{m_k} \right)$$

$$= \sum_{k=1}^{K} \sigma^2 \left( (\mu_k^A)^2 m_k - \sum_{i=1}^{m_k} \mu_{ki}^2 \right)$$

$$\sum_{k=1}^{K} \left( \sum_{i=1}^{m_k} \sum_{j=1}^{m_k} (Y_{ki} m_k - 2\mu_k^A m_k - 1) + Y_{ki} + (\mu_k^A)^2 m_k - \frac{\sum_{i=1}^{m_k} Y_{ki}^2}{m_k} + 2 \epsilon_{m_k} \{ \mu_k Y_{ki} \} - \hat{\phi}_k \right)$$

$$= \sum_{k=1}^{K} \sigma^2 \left( (\mu_k^A)^2 m_k - \hat{\phi}_k \right)$$

where $\hat{\phi}_k = \frac{\sum_{i=1}^{m_k} \mu_{ki}^2}{m_k} = \epsilon_{m_k} \{ e^{2x_i|\alpha} \}$
\[
\sum_{k=1}^{K} \left( \bar{Y}_k (Y_k m_k - 2\hat{\mu}_k^A m_k - 1) + \frac{\sum_{i=1}^{m_k} Y_{ki}^2}{m_k} + 2\varepsilon_{m_k} \{\mu_k Y_k\} + (\hat{\mu}_k^A)^2 m_k - \hat{\phi}_k \right) = \sum_{k=1}^{K} \sigma^2 (\hat{\mu}_k^A)^2 m_k - \hat{\phi}_k
\]

Due to the Bernoulli assumption on \(Y_{ki}\) then
\[
\sum_{k=1}^{K} \left( \bar{Y}_k - \frac{\sum_{i=1}^{m_k} Y_{ki}^2}{m_k} \right) = 0, \text{ so}
\]
\[
\sum_{k=1}^{K} \left( \bar{Y}_k (\bar{Y}_k m_k - 2\hat{\mu}_k^A m_k - 1) + 2\varepsilon_{m_k} \{\mu_k Y_k\} + (\hat{\mu}_k^A)^2 m_k - \hat{\phi}_k \right) = \sum_{k=1}^{K} \sigma^2 (\hat{\mu}_k^A)^2 m_k - \hat{\phi}_k
\]
\[
\sum_{k=1}^{K} \left( \varepsilon_{m_k} \{Y_k\} \varepsilon_{m_k} \{Y_k\} m_k - 2\hat{\mu}_k^A m_k - 1) + 2\varepsilon_{m_k} \{\mu_k Y_k\} + (\hat{\mu}_k^A)^2 m_k - \hat{\phi}_k \right) = \sum_{k=1}^{K} \sigma^2 (\hat{\mu}_k^A)^2 m_k - \hat{\phi}_k
\]
then
\[
\sigma^2 = \sum_{k=1}^{K} \left( \varepsilon_{m_k} \{Y_k\} \varepsilon_{m_k} \{Y_k\} m_k - 2\hat{\mu}_k^A m_k - 1) + 2\varepsilon_{m_k} \{\mu_k Y_k\} (\hat{\mu}_k^A)^2 m_k - \hat{\phi}_k \right) + 1
\]
and an unbiased estimate of \(\sigma^2\) is
\[
(\hat{\sigma}^2)_i = \frac{1}{K} \sum_{k=1}^{K} \left( \varepsilon_{m_k} \{Y_k\} \varepsilon_{m_k} \{Y_k\} m_k - 2\hat{\mu}_k^A m_k - 1) + 2\varepsilon_{m_k} \{\mu_k Y_k\} (\hat{\mu}_k^A)^2 m_k - \hat{\phi}_k \right) + 1
\]

**A.4.3 Estimation of \(\sigma^2\) in the aggregated part of the PBEE approach.**

We know
\[
E[(\bar{Y}_k^A - \hat{\mu}_k^A)^2] = \sigma^2 (\hat{\mu}_k^A)^2 - \hat{\phi}_k (n_k - m_k)^{-1}) + (\hat{\mu}_k^A - \hat{\phi}_k)(n_k - m_k)^{-1} \quad \text{for} \ k=1,\ldots,K
\]
so
\[
\sum_{k=1}^{K} (\bar{Y}_k^A - \hat{\mu}_k^A)^2 = K \sum_{k=1}^{K} \sigma^2 (\hat{\mu}_k^A)^2 - \hat{\phi}_k (n_k - m_k)^{-1}) + (\hat{\mu}_k^A - \hat{\phi}_k)(n_k - m_k)^{-1} \\
\sum_{k=1}^{K} (Y_k^A - \hat{\mu}_k^A)^2 = K \sum_{k=1}^{K} \sigma^2 (\hat{\mu}_k^A)^2 - \hat{\phi}_k (n_k - m_k)^{-1}) + (\hat{\mu}_k^A - \hat{\phi}_k)(n_k - m_k)^{-1} \\
\sum_{k=1}^{K} (\bar{Y}_k^A - \hat{\mu}_k^A)^2 - (\hat{\mu}_k^A - \hat{\phi}_k)(n_k - m_k)^{-1}) = \sigma^2 \sum_{k=1}^{K} (\hat{\mu}_k^A)^2 - \hat{\phi}_k (n_k - m_k)^{-1} \\
\]
\[ \sigma^2 = \sum_{k=1}^{K} \left( \hat{Y}_k - \hat{\mu}_k \right)^2 - \left( \hat{\mu}_k - \hat{\phi}_k \right) \left( n_k - m_k \right)^{-1} \left( \hat{\mu}_k - \hat{\phi}_k \right)^2 \left( n_k - m_k \right)^{-1} \]

and an unbiased estimate of \( \sigma^2 \) is

\[ (\hat{\sigma}^2)_A = \frac{1}{K} \sum_{k=1}^{K} \left( \hat{Y}_k - \hat{\mu}_k \right)^2 - \left( \hat{\mu}_k - \hat{\phi}_k \right) \left( n_k - m_k \right)^{-1} \left( \hat{\mu}_k - \hat{\phi}_k \right)^2 \left( n_k - m_k \right)^{-1} \]
A.5 Construction of geographics units.

Objective

The goal was to construct well-defined contiguous small-areas or zones, with an appropriate population size and the maximum level of social homogeneity. This Atlas has used the results previously obtained in the Atlas of Mortality in Spain\textsuperscript{16}.

Criteria

Three important features had to be taken into account in order to construct small-areas: availability of information, population size and social homogeneity of the areas\textsuperscript{141,142}.

1) Availability of information. In Spain, for confidentiality reasons, annual mortality data at the municipal level are available only for areas of 10,000 people or greater. However, information was available for smaller areas (i.e., at least 3,500 inhabitants) if the mortality data were aggregated for a period of three or more years.

2) Population size. Spanish municipalities are heterogeneous in terms of their socio-economic characteristics and population size. For example, regarding their population size, more than 80\% of the municipalities have fewer than 3,500 inhabitants. Thus, in order to yield reliable estimates of mortality rates, areas had to have a minimum population size.

3) Social homogeneity. Adjacent areas are often similar in terms of their social characteristics. It was possible to group municipalities with less than 3,500 inhabitants into bigger homogenous areas based on criteria of contiguity and socio-economic characteristics.
Methods

The smallest municipalities of Spain (municipalities fewer than 3,500 inhabitants) were used as the geographical building blocks to construct the zones. Information of data and demarcation lines of municipalities was provided by the Spanish Geographic National Institute. An available proxy of income level was assigned to each zone\textsuperscript{143}. The zones were aggregated automatically or by hand according to specific criteria\textsuperscript{115}. Thus, areas were aggregated automatically using an algorithm developed with a Geographic Information System program. The three specific steps followed were: 1) Small municipalities were selected for each Spanish Autonomous Community; 2) Estimated income level of all municipalities were classified into four categories: A = "Low income". Income level less than 700,001 pts, B = "Relatively low income". Income level from 700,001 to 880,000 pts, C = "Relatively high income". Income level from 880,001 to 1,100,000 pts, D = "High income". Income level more or equal than 1,100.001 pts; and 3) contiguous small areas with similar income level categories were merged automatically by using the GIS system to reach a minimum population size of 3,500 people.

Remaining areas were aggregated by hand using specific rules modified from the criteria proposed by Haining\textsuperscript{142}. The three main criteria followed where:

1) The “Island” criterion: A small municipality of one income category, entirely surrounded by a different municipality with a different income category, was absorbed into the surrounding area if that surrounding municipality had less than 10,000 people. If the population size of the surrounding municipality was greater than 10,000, the smaller area was joined using “Income level and proximity criteria”.

2) The “Neighborhood” criterion: Small municipalities can be joined with those larger neighbouring municipalities (except when they are larger than 10,000 people) having similar income levels.
3) The “Income level and proximity” criteria: Municipalities entirely surrounded by other municipalities with more than 10,000 people are joined with non-adjacent areas using the most similar income level and proximity criteria.
A.6 Life expectancy maps.


98 Spiegelhalter DJ, Best NG, Carlin BP, Van der Linde A. Bayesian deviance, the effective number of parameters and the comparison of arbitrarily complex models. *Journal of the Royal Statistical Society* B 2002a; 64:583-640.


