Clusters with unequal size: maximum likelihood versus weighted estimation in large samples

Supplementary material

HERMANS, Lisa; NASSIRI, Vahid; MOLENBERGHS, Geert; Kenward, Michael G.; VAN DER ELST, Wim; AERTS, Marc & VERBEKE, Geert (2018) Clusters with unequal size: maximum likelihood versus weighted estimation in large samples. In: Statistica sinica, 28 (3), pag. 1107-1132.

DOI: 10.5705/ss.202016.0019
Handle: http://hdl.handle.net/1942/25840

Clusters With Unequal Size: Maximum Likelihood Versus Weighted Estimation in Large Samples

Lisa Hermans[1], Vahid Nassiri[2], Geert Molenberghs[1,2],
Michael G. Kenward[3], Wim Van der Elst[4], Marc Aerts[1]
and Geert Verbeke[2,1]

[1] *I-BioStat, Universiteit Hasselt, B-3590 Diepenbeek, Belgium*

[2] *I-BioStat, KU Leuven, B-3000 Leuven, Belgium*

[3] *London, United Kingdom*

[4] *Janssen Pharmaceutica, B-2340 Beerse, Belgium*

Supplementary Material

Section S1 explains the incompleteness in the compound-symmetry model based on the characterization of Hermans *at al.* (2017). The resulting lack of closed-form solutions for MLE are outlined in Section S2 and further calculations in Section S3. Background on the pseudo-likelihood-based split-sample method is given in Section S4. More on the derivation of weights for the compound-symmetry case are given in Section S5. Section S6 and S7 give more details about respectively a first and second simulation study. Section S8 describes the use of `R` for the analysis of the case study.

S1 Incompleteness in the Compound-symmetry Model

Based on (4.1) and the multiplicity of the cluster sizes, the sufficient statistics are:

$$W_{1k} = \sum_{i=1}^{c_k}\sum_{j=1}^{n_k} Y_{ij}^{(k)}, \tag{S1.1}$$

$$W_2 = \sum_{k=1}^{L}\sum_{i=1}^{c_k}\sum_{j=1}^{n_k} \left(Y_{ij}^{(k)}\right)^2, \tag{S1.2}$$

$$W_{3k} = \sum_{i=1}^{c_k}\left(\sum_{j=1}^{n_k} Y_{ij}^{(k)}\right)^2, \tag{S1.3}$$

$$W_{4k} = c_k. \tag{S1.4}$$

The conditional and marginal expectations of (S1.1)–(S1.4) are:

$$\begin{aligned}
E(W_{1k}|c_k) &= c_k n_k \mu, \\
E(W_{1k}) &= N\mu\pi_k n_k, \\
E(W_2|c_k) &= \sum_{k=1}^{L} c_k n_k(\sigma^2 + d + \mu^2), \\
E(W_2) &= N(\sigma^2 + d + \mu^2)\sum_{k=1}^{L}\pi_k n_k, \\
E(W_{3k}|c_k) &= c_k\left\{n_k(\sigma^2 + d + \mu^2) + n_k(n_k - 1)(d + \mu^2)\right\}, \\
E(W_{3k}) &= N\pi_k n_k\left\{(\sigma^2 + d + \mu^2) + (n_k - 1)(d + \mu^2)\right\}, \\
E(W_{4k}) &= N\pi_k.
\end{aligned}$$

Group all sufficient statistics in $\boldsymbol{W}$ and define a function

$$g(\boldsymbol{w}) = \sum_{k=1}^{L} \lambda_k \frac{w_{1k}}{w_{4k}}. \tag{S1.5}$$

Then,

$$E\left\{g(\boldsymbol{W}|\boldsymbol{W}_4)\right\} = \sum_{k=1}^{L} \lambda_k \frac{E(W_{1k}|W_{4k})}{W_{4k}} = \mu \sum_{k=1}^{L} \lambda_k n_k,$$

and hence

$$E\left\{g(\boldsymbol{W})\right\} = \mu \sum_{k=1}^{K} \lambda_k n_k.$$

Thus, every solution $\boldsymbol{\lambda} \perp \boldsymbol{n}$, where $\boldsymbol{n} = (n_1, \ldots, n_K)'$, provides a counterexample, establishing incompleteness.

Such a vector $\boldsymbol{\lambda}$ exists if and only if $K \geq 2$, for which it is assumed that at least two $c_k > 0$ (i.e., at least two different cluster sizes occur).

S2 Likelihood-based Estimation of the CS Model

S2.1 Score Functions

The score function has components:

$$\frac{\partial \ell}{\partial \mu_k} = \frac{1}{\sigma_k^2 + n_k d_k} \left(\sum_{i=1}^{c_k} \sum_{j=1}^{n_1} y_{ij}^{(k)} - c_k n_k \mu_k \right), \tag{S2.1}$$

$$\begin{aligned} \frac{\partial \ell}{\partial \sigma_k^2} &= \frac{-c_k n_k}{2\sigma_k^2} \cdot \frac{\sigma_k^2 + (n_k - 1) d_k}{\sigma_k^2 + n_k d_k} + \frac{c_k n_k S_k}{2\sigma_k^4} \\ &\quad - \frac{d_k (2\sigma_k^2 + n_k d_k) c_k n_k T_k}{2\sigma_k^4 (\sigma_k^2 + n_k d_k)^2}, \end{aligned} \tag{S2.2}$$

$$\frac{\partial \ell}{\partial d_k} = \frac{-c_k n_k}{2(\sigma_k^2 + n_k d_k)} + \frac{c_k n_k T_k}{2(\sigma_k^2 + n_k d_k)^2}, \tag{S2.3}$$

with

$$S_k = \frac{1}{c_k n_k} Q_k = \frac{1}{c_k n_k} \sum_{i=1}^{c_k} \boldsymbol{Z}_i^{(k)\prime} \boldsymbol{Z}_i^{(k)}, \tag{S2.4}$$

$$T_k = \frac{1}{c_k n_k} R_k = \frac{1}{c_k n_k} \sum_{i=1}^{c_k} \boldsymbol{Z}_i^{(k)\prime} \boldsymbol{J}_{\boldsymbol{n_k}} \boldsymbol{Z}_i^{(k)}. \tag{S2.5}$$

S2.2 Lack of Closed-form Solution when $K \geq 2$

The lack of a closed form when $K \geq 2$ is well known, but we highlight a few relevant features here. More detail is given in Supplementary Materials S3. Function (4.4) can be turned into the log-likelihood kernel for the conventional situation where there is a common mean parameter and common variance components across all cluster sizes, i.e., $\ell(\mu, \sigma^2, d)$. The score functions follow from summing the terms in (S2.1)–(S2.3) across cluster sizes:

$$\frac{\partial \ell}{\partial \mu} = \sum_{k=1}^{K} \frac{\partial \ell}{\partial \mu_k}\bigg|_{\mu_k=\mu}, \qquad \frac{\partial \ell}{\partial \sigma^2} = \sum_{k=1}^{K} \frac{\partial \ell}{\partial \sigma_k^2}\bigg|_{\sigma_k^2=\sigma^2}, \qquad \frac{\partial \ell}{\partial d} = \sum_{k=1}^{K} \frac{\partial \ell}{\partial d_k}\bigg|_{d_k=d}. \tag{S2.6}$$

Solving the score equation in (S2.6) for the mean, using that

$$\Sigma_{n_k}^{-1} = \frac{1}{\sigma^2} I_{n_k} - \frac{d}{\sigma^2(\sigma^2 + n_k d)} J_{n_k},$$

leads to the identity:

$$\widehat{\mu} = \frac{\sum_{k=1}^{K} \frac{n_k c_k}{\sigma^2 + n_k d} \overline{Y}^{(k)}}{\sum_{k=1}^{K} \frac{n_k c_k}{\sigma^2 + n_k d}} = \frac{\sum_{k=1}^{K} \frac{n_k c_k}{\sigma^2 + n_k d} \widehat{\mu_k}}{\sum_{k=1}^{K} \frac{n_k c_k}{\sigma^2 + n_k d}}, \tag{S2.7}$$

where $\widehat{\mu_k}$ as in (4.5). For the variance components, only implicit identities follow; they are functions of (S2.4)–(S2.5). These take the form of high-degree polynomials, for which no general explicit solution exists. While (S2.7) is explicit, it is a weighted average of the cluster-size specific averages $\overline{Y}^{(k)}$, with weights depending on the variance components. This, combined with the result for the variance components, implies that there is no explicit solution, unless the variance components are known or the cluster size is constant.

S3 Full Likelihood

Referring to the conventional situations, i.e. $\ell(\mu, \sigma^2, d)$ in (4.2) and the score equation in (S2.6), also second derivatives can be calculated:

$$\frac{\partial^2 \ell}{\partial \mu^2} = \sum_{k=1}^{K} \frac{-c_k n_k}{\sigma^2 + n_k d} \tag{S3.1}$$

$$\frac{\partial^2 \ell}{\partial \sigma^2 \partial \mu} = \sum_{k=1}^{K} \frac{-1}{(\sigma^2 + n_k d)^2} \left(\sum_{i=1}^{c_k} \sum_{j=1}^{n_k} \boldsymbol{y}_{ij}^{(k)} - c_k n_k \mu_k \right) \tag{S3.2}$$

$$\frac{\partial^2 \ell}{\partial d \partial \mu_k} = \sum_{k=1}^{K} \frac{-n_k}{(\sigma^2 + n_k d)^2} \left(\sum_{i=1}^{c_k} \sum_{j=1}^{n_k} \boldsymbol{y}_{ij}^{(k)} - c_k n_k \mu_k \right) \tag{S3.3}$$

$$\frac{\partial^2 \ell}{\partial \mu \partial \sigma^2} = \sum_{k=1}^{K} \left(\frac{-1}{\sigma^4} + \frac{d(2\sigma^2 + n_k d) n_k}{\sigma^4 (\sigma^2 + n_k d)^2} \right) \sum_{i=1}^{c_k} \sum_{j=1}^{n_k} Z_{ij}^{(k)} \tag{S3.4}$$

$$\frac{\partial^2 \ell}{(\partial\sigma^2)^2} = \sum_{k=1}^{K} \left(\frac{c_k n_k}{2\sigma^4} \cdot \frac{\sigma^2 + (n_k - 1)d}{\sigma^2 + n_k d} - \frac{c_k n_k}{2\sigma^2} \cdot \frac{d}{(\sigma^2 + n_k d)^2} - \frac{c_k n_k S_k}{\sigma^6} \right.$$
$$\left. - d c_k n_k T_k \frac{\sigma^2(\sigma^2 + n_k d) - (2\sigma^2 + n_k d)^2}{\sigma^6(\sigma^2 + n_k d)^3} \right) \tag{S3.5}$$

$$\frac{\partial^2 \ell}{\partial d \partial \sigma^2} = \sum_{i=1}^{K} \left(\frac{c_k n_k}{2(\sigma^2 + n_k d)} - \frac{c_k n_k T_k}{\sigma^4} \cdot \frac{(\sigma^2 + n_k d)^2 - n_k d(2\sigma^2 + n_k d)}{(\sigma^2 + n_k d)^3} \right) \tag{S3.6}$$

$$\frac{\partial^2 \ell}{\partial \mu \partial d} = \sum_{k=1}^{K} \frac{-n_k}{(\sigma^2 + n_k d)^2} \sum_{i=1}^{c_k} \sum_{j=1}^{n_k} Z_{ij}^{(k)} \tag{S3.7}$$

$$\frac{\partial^2 \ell}{\partial \sigma^2 \partial d} = \sum_{k=1}^{K} \left(\frac{c_k n_k}{2(\sigma^2 + n_k d)^2} - \frac{c_k n_k T_k}{(\sigma^2 + n_k d)^3} \right) \tag{S3.8}$$

$$\frac{\partial^2 \ell}{\partial d^2} = \sum_{k=1}^{K} \left(\frac{c_k n_k^2}{2(\sigma^2 + n_k d)^2} - \frac{c_k n_k^2 T_k}{(\sigma^2 + n_k d)^3} \right). \tag{S3.9}$$

Should we use conditional likelihood, then the log-likelihood's kernel equals:

$$
\begin{aligned}
L \;\propto\; & \prod_{i=1}^{k} \frac{1}{(2\pi)^{n_1/2}|\Sigma_{n_1}|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{y}_{i1}-\boldsymbol{\mu}_{n_1})'\Sigma_{n_1}^{-1}(\boldsymbol{y}_{i1}-\boldsymbol{\mu}_{n_1})\right\} \\
& \times \left[\frac{\Phi(\alpha+\boldsymbol{y}_{i1}'\boldsymbol{\beta})}{\Phi\left(\frac{\alpha+\boldsymbol{\mu}_{n_1}'\boldsymbol{\beta}}{\sqrt{1+\boldsymbol{\beta}'\Sigma_{n_1}\boldsymbol{\beta}/n_1}}\right)}\right] \\
& \times \prod_{i=k+1}^{N} \frac{1}{(2\pi)^{n_2/2}|\Sigma_{n_2}|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{y}_{i}-\boldsymbol{\mu}_{n_2})'\Sigma_{n_2}^{-1}(\boldsymbol{y}_{i}-\boldsymbol{\mu}_{n_2})\right\} \\
& \times \left[\frac{1-\Phi(\alpha+\boldsymbol{y}_{i1}'\boldsymbol{\beta})}{1-\Phi\left(\frac{\alpha+\boldsymbol{\mu}_{n_1}'\boldsymbol{\beta}}{\sqrt{1+\boldsymbol{\beta}'\Sigma_{n_1}\boldsymbol{\beta}/n_1}}\right)}\right] \qquad \text{(S3.10)}
\end{aligned}
$$

$$
\begin{aligned}
\ell \;\propto\; & -\frac{1}{2}\sum_{i=1}^{k}\left\{\ln|\Sigma_{n_1}| + (\boldsymbol{y}_i-\boldsymbol{\mu}_{n_1})'\Sigma_{n_1}^{-1}(\boldsymbol{y}_{i1}-\boldsymbol{\mu}_{n_1})\right\} \\
& -k\ln\Phi(\tilde{\alpha}+\boldsymbol{\mu}_{n_1}'\tilde{\boldsymbol{\beta}}) \\
& -\frac{1}{2}\sum_{i=k+1}^{N}\left\{\ln|\Sigma_{n_2}| + (\boldsymbol{y}_i-\boldsymbol{\mu}_{n_2})'\Sigma_{n_2}^{-1}(\boldsymbol{y}_i-\boldsymbol{\mu}_{n_2})\right\} \\
& -(N-k)\ln\left\{1-\Phi(\tilde{\alpha}+\boldsymbol{\mu}_{n_1}'\tilde{\boldsymbol{\beta}})\right\},, \qquad \text{(S3.11)}
\end{aligned}
$$

with $\tilde{\alpha} = \frac{\alpha}{\sqrt{1+\boldsymbol{\beta}'\Sigma_{n_1}\boldsymbol{\beta}/n_1}}$ and $\tilde{\boldsymbol{\beta}} = \frac{\boldsymbol{\beta}}{\sqrt{1+\boldsymbol{\beta}'\Sigma_{n_1}\boldsymbol{\beta}/n_1}}$

The corresponding score equations are:

$$
\begin{aligned}
\frac{\partial\ell}{\partial\mu} \;=\; & \frac{1}{\sigma^2+n_1 d}\left(\sum_{i=1}^{k}\sum_{j=1}^{n_1}\boldsymbol{y}_{ij} - kn_1\mu\right) - kn_1\boldsymbol{j}_{\boldsymbol{n_1}}'\tilde{\boldsymbol{\beta}}\frac{\phi(\tilde{\alpha}+\boldsymbol{\mu}_{n_1}'\tilde{\boldsymbol{\beta}})}{\Phi(\tilde{\alpha}+\boldsymbol{\mu}_{n_1}'\tilde{\boldsymbol{\beta}})} \\
& +\frac{1}{\sigma^2+n_2 d}\left(\sum_{i=k+1}^{N}\sum_{j=1}^{n_2}\boldsymbol{y}_{ij} - (N-k)n_2\mu\right) \\
& -(N-k)n_2\boldsymbol{j}_{\boldsymbol{n_1}}'\tilde{\boldsymbol{\beta}}\frac{\phi(\tilde{\alpha}+\boldsymbol{\mu}_{n_1}'\tilde{\boldsymbol{\beta}})}{\Phi(\tilde{\alpha}+\boldsymbol{\mu}_{n_1}'\tilde{\boldsymbol{\beta}})}, \qquad \text{(S3.12)}
\end{aligned}
$$

with $\frac{\partial \ell}{\partial \sigma^2}$ and $\frac{\partial \ell}{\partial d}$ identical to (S2.2) and (S2.3). The components of the Hessian are:

$$
\begin{aligned}
\frac{\partial^2 \ell}{\partial \mu^2} &= \frac{-kn_1}{\sigma^2 + n_1 d} - \frac{(N-k)n_2}{\sigma^2 + n_2 d} \\
&\quad -kn_1 \boldsymbol{j}'_{\boldsymbol{n_1}} \tilde{\boldsymbol{\beta}} \left[\frac{-\Phi(\tilde{\alpha} + \boldsymbol{\mu}'_{n_1}\tilde{\boldsymbol{\beta}}) \cdot \phi(\tilde{\alpha} + \boldsymbol{\mu}'_{n_1}\tilde{\boldsymbol{\beta}}) \cdot (\tilde{\alpha} + \boldsymbol{\mu}'_{n_1}\tilde{\boldsymbol{\beta}}) \cdot \boldsymbol{j}'_{\boldsymbol{n_1}}\tilde{\boldsymbol{\beta}}}{\Phi^2(\tilde{\alpha} + \boldsymbol{\mu}'_{n_1}\tilde{\boldsymbol{\beta}})} \right. \\
&\quad \left. - \frac{\phi^2(\tilde{\alpha} + \boldsymbol{\mu}'_{n_1}\tilde{\boldsymbol{\beta}}) \cdot \boldsymbol{j}'_{\boldsymbol{n_1}}\tilde{\boldsymbol{\beta}}}{\Phi^2(\tilde{\alpha} + \boldsymbol{\mu}'_{n_1}\tilde{\boldsymbol{\beta}})} \right] \\
&\quad +(N-k)n_2 \boldsymbol{j}'_{\boldsymbol{n_1}} \tilde{\boldsymbol{\beta}} \\
&\quad \times \left[\frac{-(1 - \Phi(\tilde{\alpha} + \boldsymbol{\mu}'_{n_1}\tilde{\boldsymbol{\beta}})) \cdot \phi(\tilde{\alpha} + \boldsymbol{\mu}'_{n_1}\tilde{\boldsymbol{\beta}}) \cdot (\tilde{\alpha} + \boldsymbol{\mu}'_{n_1}\tilde{\boldsymbol{\beta}}) \cdot \boldsymbol{j}'_{\boldsymbol{n_1}}\tilde{\boldsymbol{\beta}}}{(1 - \Phi(\tilde{\alpha} + \boldsymbol{\mu}'_{n_1}\tilde{\boldsymbol{\beta}}))^2} \right. \\
&\quad \left. + \frac{\phi^2(\tilde{\alpha} + \boldsymbol{\mu}'_{n_1}\tilde{\boldsymbol{\beta}}) \cdot \boldsymbol{j}'_{\boldsymbol{n_1}}\tilde{\boldsymbol{\beta}}}{(1 - \Phi(\tilde{\alpha} + \boldsymbol{\mu}'_{n_1}\tilde{\boldsymbol{\beta}}))^2} \right] \qquad \text{(S3.13)} \\
\frac{\partial^2 \ell}{\partial \sigma^2 \partial \mu} &= \frac{-1}{(\sigma^2 + n_1 d)^2} \left(\sum_{i=1}^{k} \sum_{j=1}^{n_1} \boldsymbol{y}_{ij} - kn_1\mu \right) \\
&\quad - \frac{1}{(\sigma^2 + n_2 d)^2} \left(\sum_{i=k+1}^{N} \sum_{j=1}^{n_2} \boldsymbol{y}_{ij} - (N-k)n_2\mu \right) \qquad \text{(S3.14)} \\
\frac{\partial^2 \ell}{\partial d \partial \mu} &= \frac{-n_1}{(\sigma^2 + n_1 d)^2} \left(\sum_{i=1}^{k} \sum_{j=1}^{n_1} \boldsymbol{y}_{ij} - kn_1\mu \right) \\
&\quad - \frac{n_2}{(\sigma^2 + n_2 d)^2} \left(\sum_{i=k+1}^{N} \sum_{j=1}^{n_2} \boldsymbol{y}_{ij} - (N-k)n_2\mu \right) \qquad \text{(S3.15)}
\end{aligned}
$$

with $\frac{\partial^2 \ell}{\partial \mu \partial \sigma^2}$, $\frac{\partial^2 \ell}{(\partial \sigma^2)^2}$, $\frac{\partial^2 \ell}{\partial d \partial \sigma^2}$, $\frac{\partial^2 \ell}{\partial \mu \partial d}$, $\frac{\partial^2 \ell}{\partial \sigma^2 \partial d}$, and $\frac{\partial^2 \ell}{\partial (d)^2}$ indentical to (S3.4), (S3.5), (S3.6), (S3.7), (S3.8), and (S3.9).

S4 Pseudo-likelihood for Split Samples

S4.1 General Considerations

Informally, a pseudo-likelihood function is one that replaces a computationally intractable or slow-to-maximize likelihood function, with another function that still produces a consistent and asymptotically normal estimator when maximised, i.e., by setting the first derivatives of the log pseudo-likelihood equal to zero and solving the resultant pseudo score equations. An early reference is Arnold and Strauss (1991), and details on various forms of pseudo-likelihood can be found in (Molenberghs and Verbeke, 2005, Ch. 9, 12, 21, 22, 24, and 25). In the current clustered setting, the likelihood contribution of a cluster is often replaced by a product of contributions for various sub-vectors. In some cases, such a sub-vector can be conditioned upon another sub-vector.

Fieuws and Verbeke (2006) and Fieuws *et al.* (2006) used pseudo-likelihood to fit mixed models to high-dimensional multivariate longitudinal data. They supplemented the standard method with an additional device. They first replaced a set of M longitudinal sequences by the $M(M-1)/2$ longitudinal pairs. This in itself is a standard application of pseudo-likelihood. They then assumed that each pair has its own parameter vector.

Symbolically, this can be written as:

$$p\ell(\boldsymbol{\theta}) \equiv p\ell(\boldsymbol{y}_{1i}, \boldsymbol{y}_{2i}, \ldots, \boldsymbol{y}_{Mi}|\boldsymbol{\theta}) = \sum_{r<s} \ell(\boldsymbol{y}_{ri}, \boldsymbol{y}_{si}|\boldsymbol{\theta}_{rs}), \tag{S4.1}$$

where $\boldsymbol{Y}_{ri}$ is the rth sequence for subject i. In (S4.1), $\boldsymbol{\theta}$ results from stacking all $M(M-1)/2$ pair-specific parameter vectors $\boldsymbol{\theta}_{rs}$. The actual parameter vector of interest is $\boldsymbol{\theta}^*$, the set of non-redundant parameters is $\boldsymbol{\theta}$.

To obtain $\boldsymbol{\theta}^*$, Fieuws and Verbeke (2006) take averages of all available estimates for that specific parameter, implying that $\widehat{\boldsymbol{\theta}}^* = A\widehat{\boldsymbol{\theta}}$ for an appropriate linear combination matrix A. Further, combining this step with general pseudo-likelihood inference, a sandwich estimator is used:

$$\sqrt{N}(\widehat{\boldsymbol{\theta}}^* - \boldsymbol{\theta}^*) = \sqrt{N}(A\widehat{\boldsymbol{\theta}} - A\boldsymbol{\theta}) \overset{\text{approx.}}{\sim} N(\mathbf{0}, AI_0^{-1}I_1I_0^{-1}A'), \tag{S4.2}$$

where

$$I_0(\boldsymbol{\theta}) = E\left[\frac{\partial^2 p\ell(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}'\partial\boldsymbol{\theta}}\right], \qquad I_1(\boldsymbol{\theta}) = E\left[\left(\frac{\partial p\ell(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}\right)' . \frac{\partial p\ell(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}\right]. \tag{S4.3}$$

Molenberghs *et al.* (2011) took a very similar route to partition a potentially large sample into sub-samples.

To fix ideas, consider log-likelihood (4.4). When used as an instrument to estimate a single vector (μ, σ^2, d), this function can be viewed a pseudo-likelihood. This setting can be generalized by assuming that a dataset, consisting of repeated measures per subject, is divided into K subgroups,

each containing c_k independent replicates. Consider the pseudo-likelihood:

$$p\ell(\boldsymbol{\theta}) = \sum_{k=1}^{K} \ell(\boldsymbol{\theta}_k | \boldsymbol{y}_1^{(k)}, \ldots, \boldsymbol{y}_{c_k}^{(k)}). \tag{S4.4}$$

While the underlying principle is similar to (S4.1), it is not identical. The similarities are: (1) all $\boldsymbol{\theta}_k$ are assumed to be different, allowing for separate, even parallel, estimation; (2) $\boldsymbol{\theta}$ stacks all vectors $\boldsymbol{\theta}_k$; (3) the parameter of interest $\boldsymbol{\theta}^*$, is found from an appropriate combination of the $\boldsymbol{\theta}_k$. Parallel estimation was also followed by Scott *et al.* (2013) and Neiswanger, Wang, and Xing (2013).

There are important differences, however. Here, and in the remainder of the article, we assume that $\ell(\boldsymbol{\theta}_k | \boldsymbol{y}_1^{(k)}, \ldots, \boldsymbol{y}_{c_k}^{(k)})$ is the likelihood that we would have, should group k be the only one in the data. That is, the individual likelihood contributions are not altered, rather the data are partitioned. This is similar to the independent partitioning done by Molenberghs *et al.* (2011). In line with their derivations, (S4.2) can also be used here. Given that $\ell_k(\boldsymbol{\theta}_k)$ is a genuine likelihood, its contributions to $I_0(\boldsymbol{\theta})$ and $I_1(\boldsymbol{\theta})$ are identical, up to the sign. As a result, $I_0(\boldsymbol{\theta})^{-1} I_1(\boldsymbol{\theta}) I_0(\boldsymbol{\theta})^{-1} = -I_0(\boldsymbol{\theta})^{-1}$, a block-diagonal matrix with blocks of the form $I_0(\boldsymbol{\theta}_k)$. We now turn to the split-sample case.

S4.2 Pseudo-likelihood for Split Sample

Molenberghs *et al.* (2011) chose

$$A = \frac{1}{K}(I, \ldots, I) \tag{S4.5}$$

to pass from $\boldsymbol{\theta}$ to $\boldsymbol{\theta}^*$. This is a sensible choice in the i.i.d. setting (e.g., when all clusters in a CS model have the same size) and with the same number of subjects per sub-sample. The estimator and precision estimator then become:

$$\widehat{\boldsymbol{\theta}}^* = \frac{1}{K}\sum_{k=1}^{K}\widehat{\boldsymbol{\theta}}_k, \tag{S4.6}$$

$$\mathrm{var}(\widehat{\boldsymbol{\theta}}^*) = \frac{1}{K}H_{\widehat{\theta}}^{-1}, \tag{S4.7}$$

with $H_{\widehat{\theta}}^{-1} = -I_0(\boldsymbol{\theta}_k)$. In this special case, the expected information matrices are identical. Alternatively, one can use the observed information matrices, and then use instead:

$$\frac{1}{K^2}\sum_{k=1}^{K}\mathcal{H}_{\widehat{\theta},k}^{-1}. \tag{S4.8}$$

where $\mathcal{H}_{\widehat{\theta},k}$ is the observed information for sub-sample k.

In this particular case, pseudo-likelihood produces the same estimator as full likelihood. This stems from the fact that all subjects follow the same distribution, in contrast to, for example, the setting set out at the start of Section 4. Should the subjects have identical distributions, but with

c_k variables, the above results can be modified accordingly. For example, (S4.6) would be replaced by

$$\widehat{\boldsymbol{\theta}}^* = \sum_{k=1}^{K} \frac{c_k}{N} \widehat{\boldsymbol{\theta}}_k,$$

with similar modification to precision estimation. In this case, full likelihood would be obtained.

In the next section, we will consider the more general case where different subjects may have a different distribution such as, for example, the CS case with a different number of measurements per cluster.

S5 Derivation of Optimal Scalar Weights for Compound-symmetry Case

To find the optimal scalar weight with minimum variance for μ we use the method of Lagrange with the constraint that the weights a_k need to sum to 1:

$$Q = \sum_{k=1}^{K} a_k^2 \frac{\sigma^2 + n_k d}{c_k n_k} - \lambda \left(\sum_{k=1}^{K} a_k - 1 \right). \tag{S5.1}$$

Solving the first partial derivative we become an expression for a_k involving λ. Summing this one produces an expression for λ, leading to the complete

formula for a_k. Precisely, the system of equations is:

$$\begin{aligned}\frac{\partial Q}{\partial a_k} = 2a_k\frac{\sigma^2+n_kd}{c_kn_k} - \lambda &= 0,\\ \frac{\partial Q}{\partial \lambda} = \sum_{k=1}^{K} a_k - 1 &= 0,\end{aligned}$$

or, alternatively:

$$\begin{aligned}a_k &= \frac{\lambda}{2}\frac{c_kn_k}{\sigma^2+n_kd},\\ \lambda &= \left(\frac{1}{2}\sum_{k=1}^{K}\frac{c_kn_k}{\sigma^2+n_kd}\right)^{-1},\end{aligned}$$

and hence

$$a_k = \frac{\frac{c_kn_k}{\sigma^2+n_kd}}{\sum_{m=1}^{K}\frac{c_mn_m}{\sigma^2+n_md}}.$$

In the same manner, expressions for b_k and g_k can be found, as we will show next. For σ^2:

$$Q = 2\sigma^4\sum_{k=1}^{K} b_k^2\frac{1}{c_k(n_k-1)} - \lambda\left(\sum_{k=1}^{K} b_k - 1\right),$$

producing the system:

$$\begin{aligned}\frac{\partial Q}{\partial b_k} = \frac{4\sigma^4 b_k}{c_k(n_k-1)} - \lambda &= 0,\\ \frac{\partial Q}{\partial \lambda} = \sum_{k=1}^{K} b_k - 1 &= 0,\end{aligned}$$

which can be rewritten as:

$$\begin{aligned}4\sigma^4 b_k &= \lambda c_k(n_k-1),\\ \lambda &= \frac{4\sigma^4}{\sum_{k=1}^{K} c_k(n_k-1)},\end{aligned}$$

and finally:

$$b_k = \frac{c_k(n_k - 1)}{\sum_{m=1}^{K} c_m(n_m - 1)}.$$

For d, the objective function is:

$$Q = \sum_{k=1}^{K} g_k^2 v_k - \lambda(\sum_{k=1}^{K} g_k - 1),$$

with

$$v_k = \frac{2}{c_k n_k}\left(\frac{\sigma^4}{n_k - 1} + 2d\sigma^2 + n_k d^2\right).$$

The system of equations now is:

$$\begin{aligned} \frac{\partial Q}{\partial g_k} = 2g_k v_k - \lambda &= 0, \\ \frac{\partial Q}{\partial \lambda} = \sum_{k=1}^{K} g_k - 1 &= 0, \end{aligned}$$

leading to:

$$\begin{aligned} g_k &= \lambda \frac{1}{2v_k}, \\ \lambda &= \frac{2}{\sum_{k=1}^{K} \frac{1}{v_k}}, \end{aligned}$$

giving the solution:

$$g_k = \frac{\frac{c_k n_k}{\frac{\sigma^4}{n_k - 1} + 2d\sigma^2 + n_k d^2}}{\sum_{m=1}^{K} \frac{c_m n_k}{n_m - 1} + 2d\sigma^2 + n_m d^2}.$$

S5.1 Cluster-by-cluster Analysis

We study the case of the most extreme partitioning, i.e., where each of the clusters is analyzed separately. This can be relevant in cases with perhaps a

limited number of very to extremely large clusters. This means that $c_k \equiv 1$ throughout. Clearly, the n_k will then no longer be unique. We will examine this case in detail, and contrast a first weighted estimator with an *ad hoc* one.

The Weighted Estimator for the Cluster-by-cluster Case

The estimator follows from setting $c_k \equiv 1$ and hence $K \equiv N$ throughout. For example, this special case can easily be considered for all expressions in Sections 6.1.1-6.1.3. Because c_k enters the inverse of the variance-covariance matrix multiplicatively, as is seen from (4.8)-(4.9), the optimal estimator that is obtained when each cluster is considered to be its own stratum, is identical to the one obtained when strata are defined in terms of all clusters of a given size. The same is true for the scalar weights.

It is insightful to consider in more detail the special case where further

the cluster sizes are all identical to n. One then easily obtains:

$$\widehat{\mu} = \frac{1}{Nn}\sum_{i=1}^{N}\sum_{j=1}^{n} Y_{ij}, \tag{S5.2}$$

$$\begin{aligned}\widehat{\sigma}^2 &= \frac{1}{Nn(n-1)}\left(n\sum_{i=1}^{N}\boldsymbol{Z}_i'\boldsymbol{Z}_i - \sum_{i=1}^{N}\boldsymbol{Z}_i'J_n\boldsymbol{Z}_i\right)\\ &= \frac{1}{Nn(n-1)}(nQ-R),\end{aligned} \tag{S5.3}$$

$$\begin{aligned}\widehat{d} &= \frac{1}{Nn(n-1)}\left(\sum_{i=1}^{N}\boldsymbol{Z}_i'J_n\boldsymbol{Z}_i - \sum_{i=1}^{N}\boldsymbol{Z}_i'\boldsymbol{Z}_i\right)\\ &= \frac{1}{Nn(n-1)}(Q-R),\end{aligned} \tag{S5.4}$$

with obvious notation for Q and R, inspired by (S2.4)–(S2.5). The corresponding variance-covariance elements, similar in spirit to (4.8)-(4.9), are:

$$\mathrm{var}(\widehat{\mu}) = \frac{\sigma^2+nd}{Nn}, \tag{S5.5}$$

$$\mathrm{var}\begin{pmatrix}\widehat{\sigma^2}\\ \widehat{d}\end{pmatrix} = \begin{pmatrix}\frac{2\sigma^4}{N(n-1)} & -\frac{2\sigma^4}{Nn(n-1)}\\ -\frac{2\sigma^4}{Nn(n-1)} & \frac{2}{Nn}\left[\frac{\sigma^4}{n-1}+2\sigma^2 d+nd^2\right]\end{pmatrix}. \tag{S5.6}$$

This estimator coincides with the MLE, as is known from Molenberghs *et al.* (2011).

A Two-stage Estimator for Compound Symmetry

In linear mixed models, there is a method of estimation, sometimes called the two-stage approach (Laird and Ware, 1982; Verbeke and Molenberghs, 2000), in which each cluster is analyzed separately to begin with, using lin-

ear regression, after which the cluster-specific parameters are summarized into fixed effects. Although the above cluster-by-cluster analysis is superficially similar to this, it is not equivalent. In particular, there is no bias (as can be seen in the two stage method), and the maximum likelihood estimator is recovered.

This approach is most useful when cluster sizes are not constant, and in models that are more complex than compound symmetry. However, to gain some insight, we develop the details of the method for the CS model with constant cluster size.

For the mean, (S5.2) is retained, as the average of the cluster-specific averages $\overline{Y}_i$. Further, define:

$$s^2 = \frac{1}{Nn}\sum_{i=1}^{N}\sum_{j=1}^{n}(Y_{ij}-\overline{Y}_i)^2, \tag{S5.7}$$

$$t^2 = \frac{1}{N}\sum_{i1}^{N}(\overline{Y}_i-\widehat{\mu})^2. \tag{S5.8}$$

Straightforward algebra shows:

$$E(s^2) = \frac{n-1}{n}\sigma^2, \tag{S5.9}$$

$$E(t^2) = \frac{N-1}{N}\left(d+\frac{1}{n}\sigma^2\right). \tag{S5.10}$$

Should n and N approach infinity, then it follows that s^2 and t^2 are asymptotically unbiased estimators for σ^2 and d, respectively. However, this is not always reasonable. In applications such as the NTP data (Section 2),

it is fair to say that the cluster size has a biological upper limit. In other situations, however, such as meta-analyses, it is sensible to assume that both n and N approach infinity.

In the next section, we will study the consequences of removing the bias. For now, a small, obvious modification is:

$$s_*^2 = \frac{n}{n-1}s^2, \tag{S5.11}$$

$$t_*^2 = \frac{N}{N-1}t^2. \tag{S5.12}$$

Now, s_*^2 is unbiased, while $E(t_*^2) = d + \sigma^2/n$, the bias σ^2/n can be made to disappear asymptotically provided it is sensible to let n grow large.

It is of interest to consider the variance-covariance structure of the estimators s^2, t^2, s_*^2, and t_*^2, as well as to make relative efficiency considerations. This will be done next.

Connections Between Estimators

Comparing algebraic expressions (S5.3)–(S5.4) with (S5.7)–(S5.8), leads to the linear relationships:

$$s^2 = \frac{n-1}{n}\widehat{\sigma^2} + 0 \cdot \widehat{d}, \tag{S5.13}$$

$$t^2 = \frac{N-1}{Nn}\widehat{\sigma^2} + \frac{N-1}{N}\widehat{d}. \tag{S5.14}$$

Relationships (S5.13)–(S5.14) can be combined with (S5.6) to produce:

$$\operatorname{var}\begin{pmatrix} s^2 \\ t^2 \end{pmatrix} = \begin{pmatrix} \frac{2(n-1)\sigma^4}{Nn^2} & 0 \\ 0 & \frac{2(N-1)^2}{N^2 n}\left[\frac{\sigma^4}{n} + 2\sigma^2 d + nd^2\right] \end{pmatrix} \tag{S5.15}$$

and, similarly,

$$\operatorname{var}\begin{pmatrix} s_*^2 \\ t_*^2 \end{pmatrix} = \begin{pmatrix} \frac{2\sigma^4}{N(n-1)} & 0 \\ 0 & \frac{2}{Nn}\left[\frac{\sigma^4}{n} + 2\sigma^2 d + nd^2\right] \end{pmatrix}. \tag{S5.16}$$

From its definition it follows that $s_*^2 \equiv \widehat{\sigma^2}$. The same is not true for t_*^2. One reason to consider it nevertheless is its independence from s_*^2. Indeed, $(\widehat{\mu}, s_*^2 \equiv \widehat{\sigma^2}, t_*^2)'$ is an estimator with mutually independent components. While the same is true when s^2 and t^2 are used instead, the biases are larger.

For this case then, the choice between $\widehat{d}$ and t_*^2 is in terms of a trade-off between efficiency and independence.

To gauge the efficiency loss when using t_*^2, the mean squared error is:

$$MSE(t_*^2) = \frac{2}{Nn}\left(\frac{\sigma^4}{n} + 2\sigma^2 d + nd^2\right) + \frac{1}{n^2}\sigma^4,$$

and hence the relative MSE:

$$RMSE(t^2; \widehat{d}) = \frac{2\left(\frac{\sigma^4}{n} + 2\sigma^2 d + nd^2\right) + \frac{N}{n}\sigma^4}{2\left(\frac{\sigma^4}{n-1} + 2\sigma^2 d + nd^2\right)}, \tag{S5.17}$$

which approaches infinity when N does, while n would remain constant.

In other words, this estimator is inconsistent unless it is being applied in situations where n can also be considered to be large.

There are three distinct situations. First, when $N/n = \lambda n + o(n)$, for some λ, i.e., when N is of the order of n^2, then, based on (S5.17), the ARE is $2(d^2 + \lambda\sigma^4)/[2d^2]$. The magnitude of the efficiency loss depends on sizes of the parameters involved. Second, when $\mathcal{O}(N) < \mathcal{O}(n^2)$, the ARE equals 1. This includes the cases where N is constant, $N = n^{1/2}$, $N = n$, and $N = n^{3/2}$, for example. A constant or slowly increasing N is plausible in a meta-analytic context. Third, if N/n increases too quickly, i.e., $\mathcal{O}(N/n) > \mathcal{O}(n)$, then the estimator t_*^2 is inconsistent. This is the case, in particular, for bounded n.

The estimators s^2 and t^2 can be combined linearly to produce unbiased estimators. In other words, based on (S5.9)–(S5.10), the following corrections can be applied to (S5.7)–(S5.8):

$$s^2_{\text{corr}} = \frac{n}{n-1}s^2, \tag{S5.18}$$

$$t^2_{\text{corr}} = \frac{N}{N-1}t^2 - \frac{N}{(n-1)(N-1)}s^2. \tag{S5.19}$$

Interestingly, this requirement reproduces (S5.13)–(S5.14): the requirement of an unbiased estimator reproduces $\widehat{\sigma^2}$ and $\widehat{d}$, presented in (S5.3)–(S5.4) and hence also with their variance.

S6 Details About the First Simulation Study

The simulation study, summarized in Section 7, is described in detail here.

S6.1 Simulation Method

The design of the simulation study is as follows.

- Each generated set of data consists of c_k clusters of size n_k, for $k = 1, \ldots, K$. We choose $K = 4$ throughout.
- For a generated set of data, the splitting is done by placing all clusters of a given size in one sub-sample.
- The CS model parameters are $\mu = 0$, $d = 1$, and $\sigma^2 = 2$.
- After estimating the three model parameters within each sub-sample, they are combined using the following weighting methods: (a) equal, (b) proportional, where the weights are

  $$w_k = \frac{c_k}{\sum_{\ell=1}^{4} c_\ell},$$

  and (c) size-proportional, where the weights for μ and d are:

  $$w_k \frac{c_k n_k}{\sum_{\ell=k}^{4} c_\ell n_\ell},$$

  while for σ^2 we take:

  $$w_k = \frac{c_k(n_k - 1)}{\sum_{\ell=1}^{4} c_\ell(n_\ell - 1)}.$$

- Per setting, 100 replications are considered.

These settings are applied to various combinations of the n_k and c_k, now described in turn.

S6.2 Setting 1: Equal $c_k \cdot n_k$, Different c_k and n_k.

Consider 150 samples in each split, as follows: $(c_1, n_1) = (3, 50)$, $(c_2, n_2) = (5, 30)$, $(c_3, n_3) = (10, 15)$, and $(c_4, n_4) = (15, 10)$. The results are presented in Table 1. Graphical depictions can be found in Figures 1 and 2. Figure 1 shows that there is a different amount of information in the various subsamples. This is not a problem, rather a consequence of the way the splits are created and the different amounts of information carried in each. It reminds us that we need to be judicious how the information from the splits will be weighted. It is not a surprise that equal weights are a poor choice. The other methods perform similarly, and all do very well. To varying degrees, the same will be seen in Settings 2 and 3.

S6.3 Setting 2: Different $c_k \cdot n_k$, Equal c_k, Different n_k

To see the effect of split size, the following choices are made: $(c_1, n_1) = (4, 25)$, $(c_2, n_2) = (4, 50)$, $(c_3, n_3) = (4, 125)$, and $(c_4, n_4) = (4, 250)$. As a consequence, the size of the splits will be 100, 200, 500, and 1000, respectively. Table 2 summarized the results, with graphical displays presented

Table 1: *First simulation study. Setting 1. Average of split-specific and combined (weighted) parameters and their precision estimates.*

	μ	var(μ)	d	var(d)	σ^2	var(σ^2)
split1	-0.00396	0.05779	0.68143	0.41395	1.98676	0.00296
split2	0.05697	0.03071	0.80997	0.19712	1.98578	0.00304
split3	-0.02111	0.01174	0.95161	0.07869	1.97690	0.00319
split4	0.01123	0.00626	0.98870	0.04677	1.98056	0.00347
Equal	0.01078	0.03769	0.85793	0.09988	1.98250	0.01406
Prop	0.00698	0.03230	0.92245	0.08568	1.98081	0.01907
Size prop	0.01078	0.03769	0.85793	0.09988	1.98260	0.01405
Full	0.00780	0.03513	0.98016	0.08614	1.98257	0.01392

Table 2: *First simulation study. Setting 2. Average of split-specific and combined (weighted) parameters and their precision estimates.*

	μ	var(μ)	d	var(d)	σ^2	var(σ^2)
split1	-0.02515	0.05227	0.83440	0.52423	2.00347	0.00730
split2	0.01287	0.05157	0.86891	0.57904	1.97285	0.00160
split3	0.06812	0.03586	0.74147	0.23681	2.00165	0.00026
split4	-0.03676	0.02979	0.68241	0.14117	1.99216	0.00006
Equal	0.00477	0.05111	0.78180	0.14770	1.99253	0.00935
Prop	0.00477	0.05111	0.78180	0.14770	1.99253	0.00935
Size prop	-0.00147	0.07139	0.72798	0.16585	1.99328	0.00447
Full	0.00530	0.06339	0.89599	0.14604	1.99333	0.00446

in Figures 3 and 4.

S6.4 Setting 3: Different $c_k \cdot n_k$, Different c_k, Equal n_k

We now choose: $(c_1, n_1) = (10, 20)$, $(c_2, n_2) = (20, 20)$, $(c_3, n_3) = (50, 20)$, and $(c_4, n_4) = (100, 20)$. Table 3 summarizes the results. Graphs can be found in Figures 5 and 6.

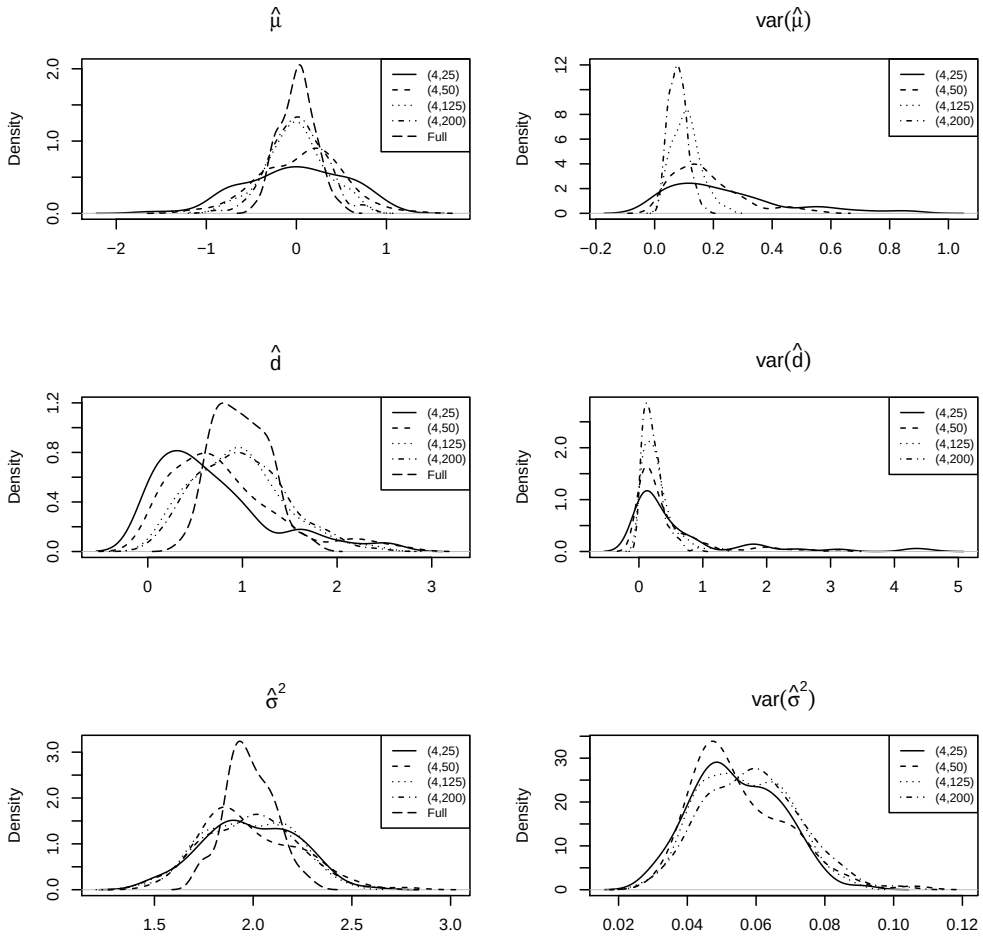

Figure 1: *First simulation study. Setting 1. Split-specific results.*

S6.5 Optimal, Approximate Optimal, and Iterated Optimal Weights

Optimal weights were discussed in Section 6.1.1. When we plug the MLE's into the optimal weights, the result of using these weights is the MLE's itself. Of course, this is a circular reasoning, which is why one needs to resort to, for example, the approximate or iterated optimal weights derived in Sec-

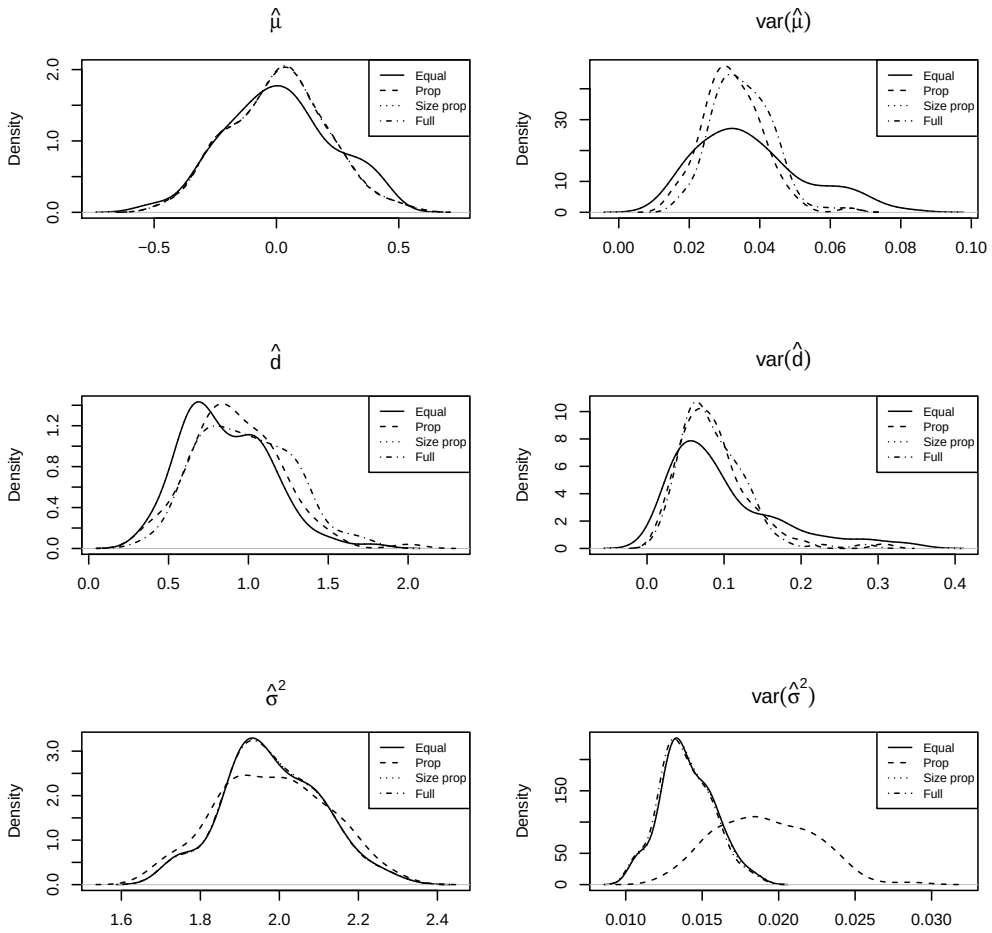

Figure 2: *First simulation study. Setting 1. Combining the results from the four splits, using equal, proportional, and size proportional weights. This is compared with full maximum likelihood.*

tion 6.1.2. For both of these, using Settings 1–3, we conducted simulations. They are reported in Figures 7–9.

It is noteworthy that the behavior of the iterated optimal weights depends on c_k and n_k. First, they often but not always converge in a single

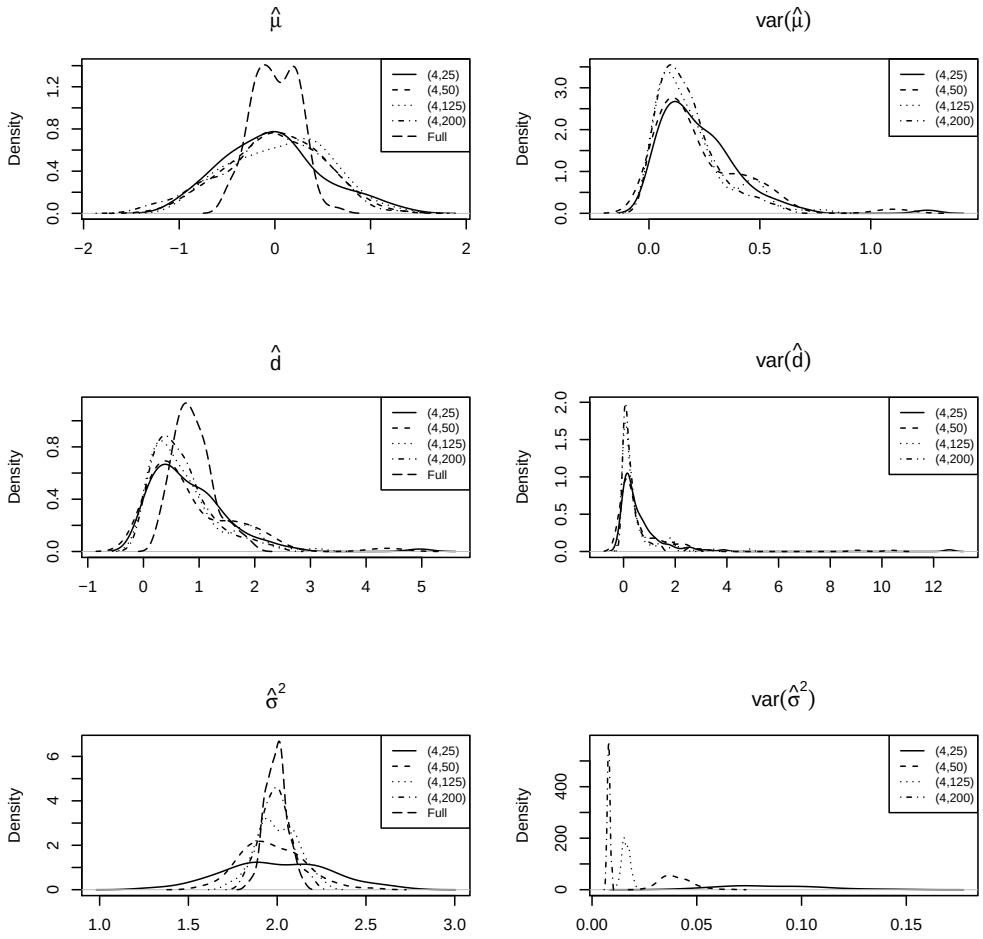

Figure 3: *First simulation study. Setting 2. Split-specific results.*

iteration; the maximum number of iterations observed in our simulations being 6. Second, the iterated optimal weights converge to size optimal weights for σ^2 and to proportional weights for d.

Taken together, it follows that both approximately optimal and iterated

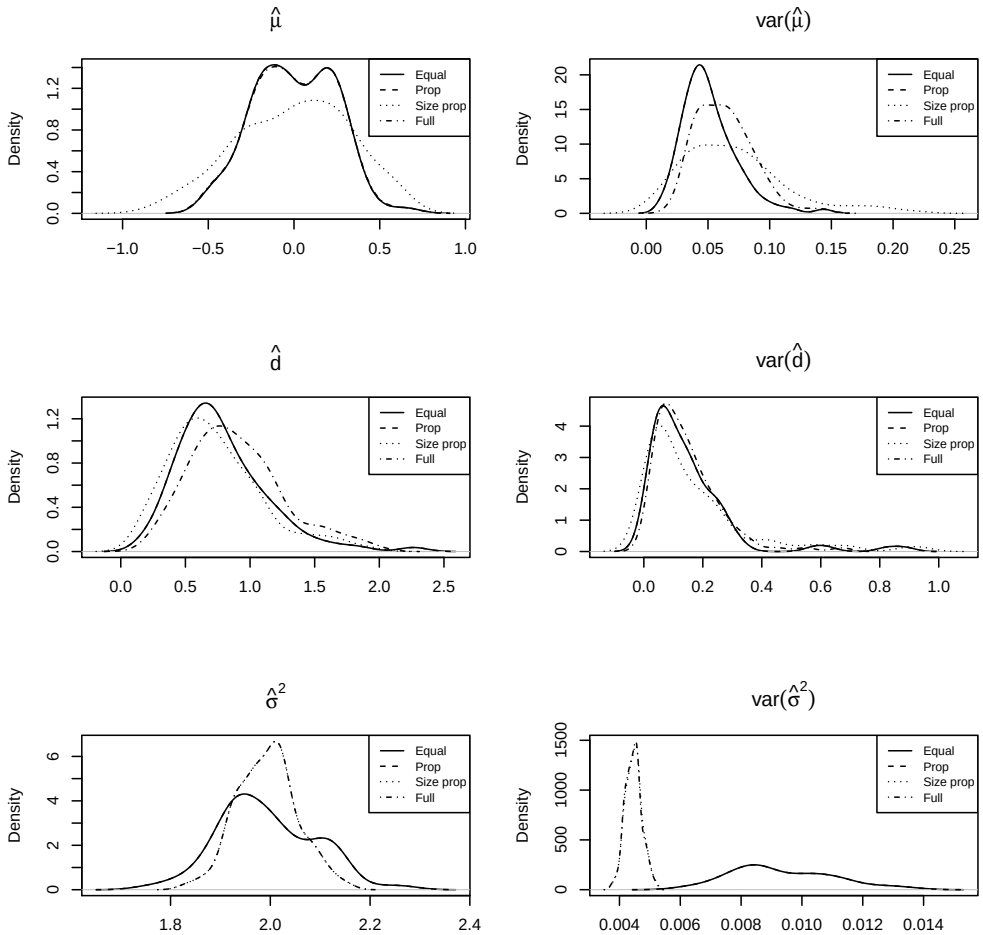

Figure 4: *First simulation study. Setting 2. Combining the results from the four splits, using equal, proportional, and size proportional weights. This is compared with full maximum likelihood.*

optimal weights provide excellent results. The specific attraction of the approximate optimal weights is that they obviate the need for iteration, which is a factor of stability and speed.

Table 3: *First simulation study. Setting 3. Average of split-specific and combined (weighted) parameters and their precision estimates.*

	μ	var(μ)	d	var(d)	σ^2	var(σ^2)
split1	0.00343	0.00900	0.84739	0.05169	2.02445	0.00190
split2	0.02553	0.00304	1.00224	0.01754	2.01962	0.00047
split3	-0.00010	0.00045	0.95794	0.00212	1.99765	0.00007
split4	0.01151	0.00012	1.01226	0.00064	1.98944	0.00002
Equal	0.01009	0.01139	0.95496	0.02694	2.00779	0.00486
Prop	0.00939	0.00604	0.98690	0.01369	1.99702	0.00234
Size prop	0.00939	0.00604	0.98690	0.01369	1.99702	0.00234
Full	0.00939	0.00614	1.00487	0.01372	1.99702	0.00233

S7 Details About the Second Simulation Study

The aim of this study is to compare the proposed method to two alternatives:

1. full maximum likelihood;
2. the proposed sample-splitting method, allowing for closed forms;
3. using multiple imputation (MI) first, to render the clusters of equal sizes, and then apply closed-form solutions to the augmented balanced data, together with the combination rules.

S7.1 Simulation Plan

In order to study the effect of cluster sizes (n_k) and number of clusters of each size (c_k), 5 different configurations are considered:

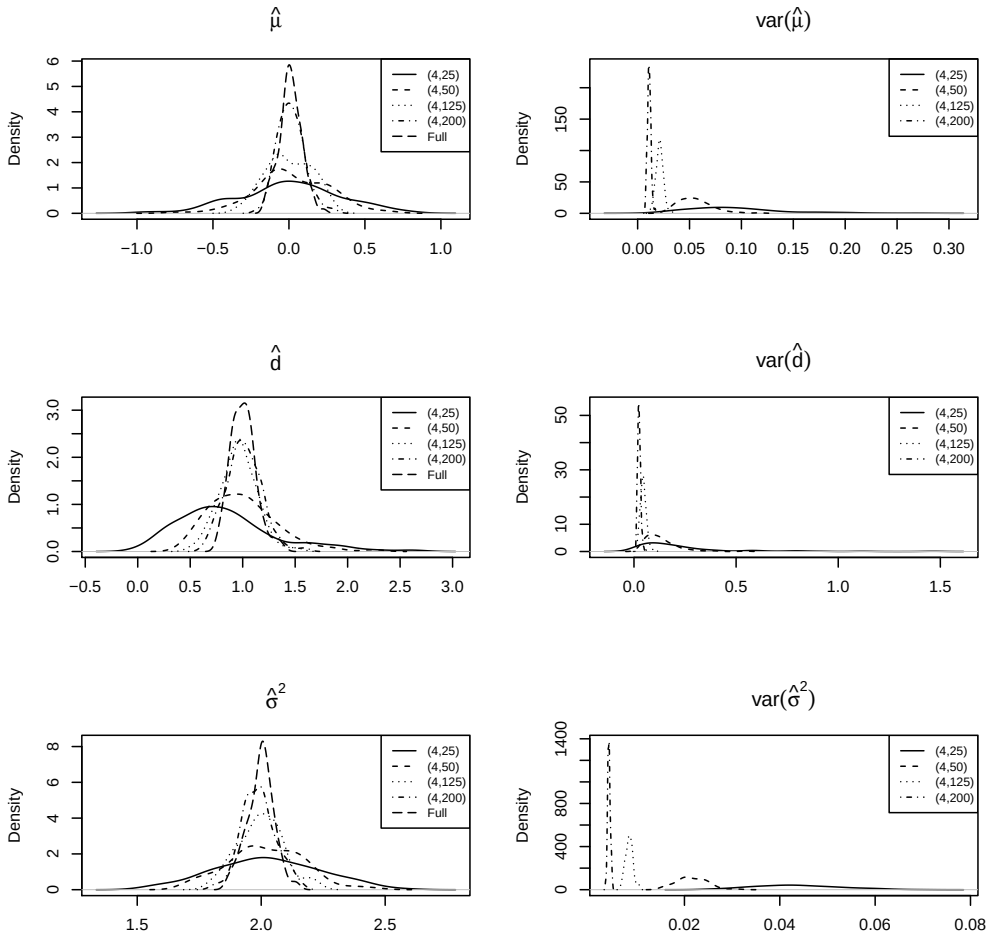

Figure 5: *First simulation study. Setting 3. Split-specific results.*

Config. 1. $c_k = (15, 25, 30, 20, 10)$, $n_k = (8, 5, 3, 9, 15)$;

Config. 2. $c_k = (150, 250, 300, 200, 100)$, $n_k = (8, 5, 3, 9, 15)$;

Config. 3. $c_k = (1500, 2500, 3000, 2000, 1000)$, $n_k = (8, 5, 3, 9, 15)$;

Config. 4. $c_k = (15, 25, 30, 20, 10)$, $n_k = (80, 50, 30, 90, 150)$;

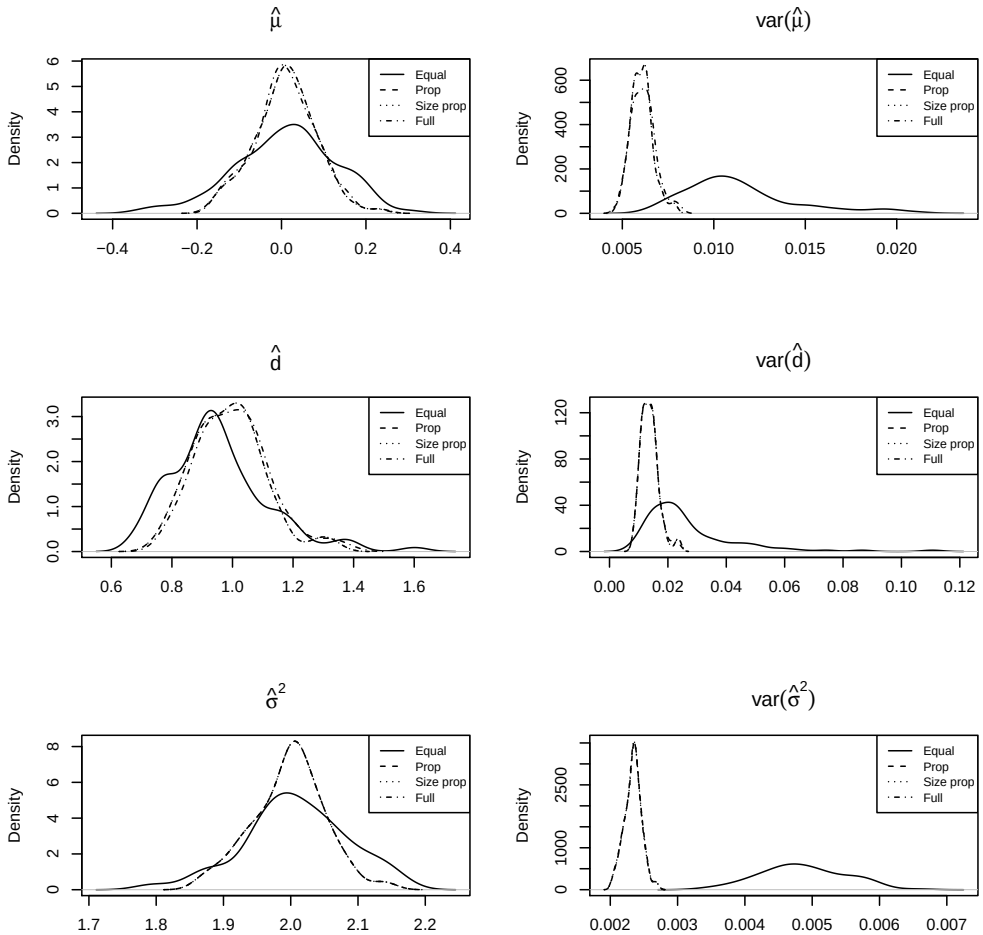

Figure 6: *First simulation study. Setting 3. Combining the results from the four splits, using equal, proportional, and size proportional weights. This is compared with full maximum likelihood.*

Config. 5. $c_k = (15, 25, 30, 20, 10)$, $n_k = (800, 500, 300, 900, 1500)$.

Each configuration is repeated 100 times.

Each cluster is generated from a CS model with $\mu_0 = 0$, $d_0 = 1$, and $\sigma_0^2 = 4$.

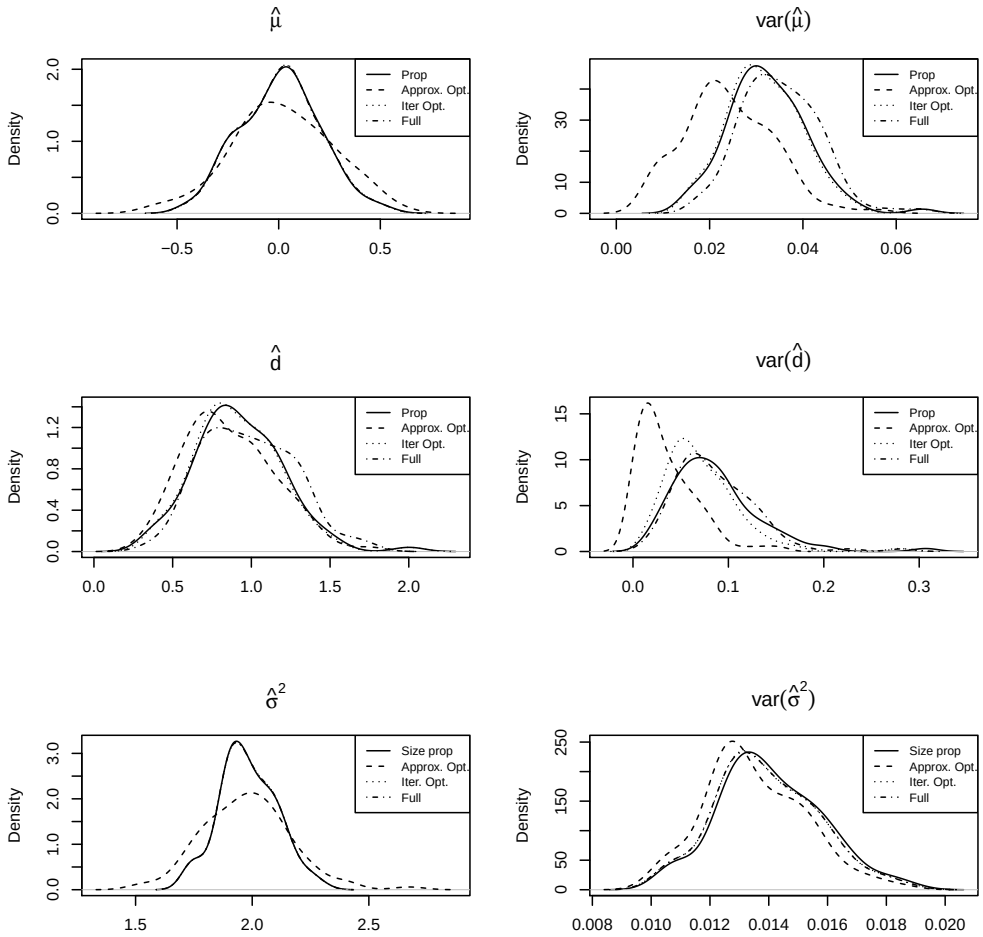

Figure 7: *First simulation study. Setting 1. (Size) proportional, approximate, and iterated optimal weights, as well as full maximum likelihood.*

For estimating the parameters using the full unbalanced data, PROC MIXED in SAS (Version 9.4) is used with the covariance structure in the REPEATED statement set to `type=cs`.

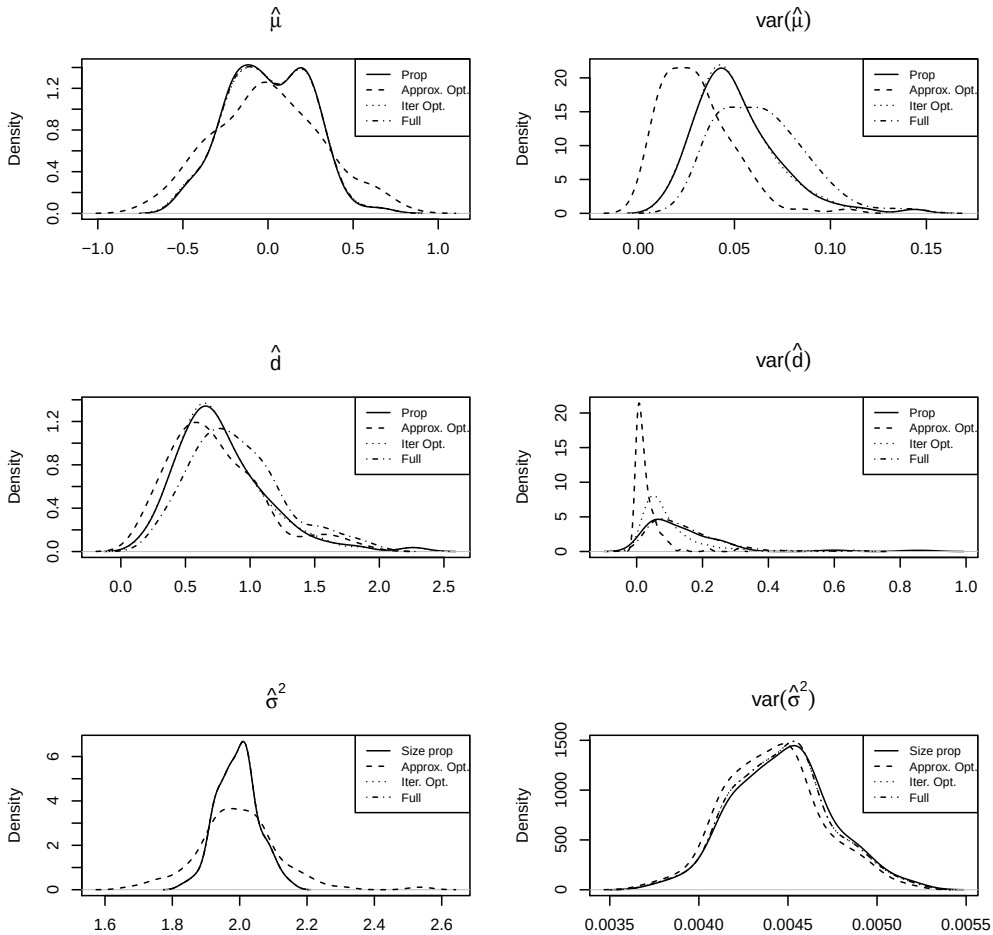

Figure 8: *First simulation study. Setting 2. (Size) proportional, approximate, and iterated optimal weights, as well as full maximum likelihood.*

The closed form solutions and their variances are implemented in `R` in three different ways. First, the formulas are implemented directly using 'for' loops. Following the ideas in Sikorska *et al.* (2013), it might be faster to replace 'for' loops with vectorized computation. For $\widehat{\mu}_k$ it is straightfor-

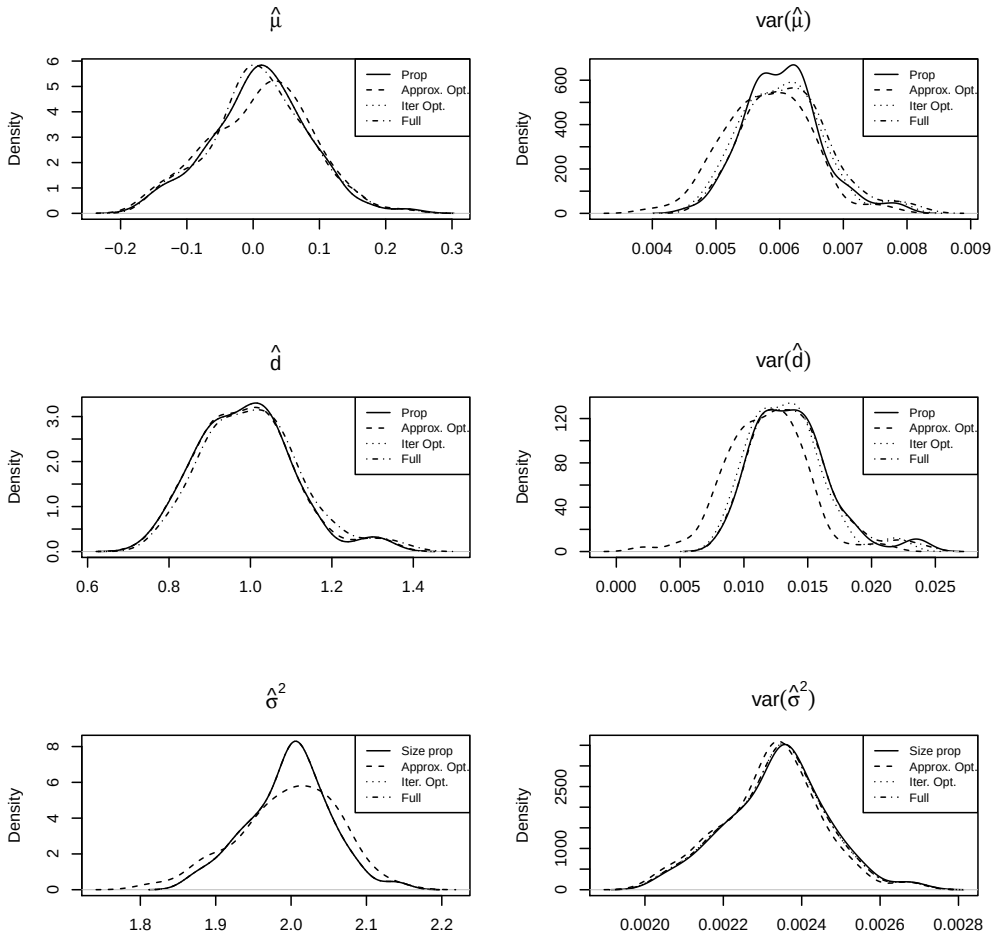

Figure 9: *First simulation study. Setting 3. (Size) proportional, approximate, and iterated optimal weights, as well as full maximum likelihood.*

ward, since one just needs to compute an arithmetic average. If Z is a n_k times c_k matrix with its ith column defined as $Z_i^{(k)} = \left(Y_i^{(k)} - \mu_k \mathbf{1}_{n_k}\right)$, then computing $\sum_{i=1}^{c_k} Z_i^{(k)\prime} Z_i^{(k)}$ is equivalent to replacing each element in matrix Z by its square, and then sum over the sum of its columns. Furthermore,

$J_{n_k} Z_i^{(k)}$ would simply compute the sum of columns in matrix Z. Therefore, $\sum_{i=1}^{c_k} Z_i^{(k)\prime} J_{n_k} Z_I^{(k)}$ is equivalent to post-multiplying Z by the sum of its columns and then sum over this vector. In this way, within each split, the parameters can be estimated avoiding 'for' loops.

Second, it is also possible to find the estimates for all of the splits at once instead of computing them separately.

A third way consists of calculating all the estimates together and not split by split in a 'for' loop. This approach is possible via imposing balance through adding missing values in the matrix but, when multiplying and summing, ignoring the missing values. This is very easy in `R`.

We will compare computation time between these three approaches, for the five configurations.

Additionally, to combine the results from sample splitting, the same weights as used in the case study are considered here as well: equal, proportional, approximate scalar, scalar, and approximate optimal. In the case of the approximate optimal weights both simple and proper variances are calculated.

For the multiple imputation based approach, $M = 20$ imputations are considered and the conventional combination rules applied.

Note that the MI approach cannot be used with configurations 1, 4, and 5, because the number of available subjects in the observed dataset is less than the number of repeated measurements, leading to a singular covariance matrix. From the remaining configurations 2 and 3, we have chosen #2, which implies smaller numbers and hence is more challenging.

For each configuration we report three results: the estimated parameters, their standard errors, and the mean square error (MSE). Furthermore, we report computation time.

S7.2 Simulation results

Table 4: *Second simulation study. Mean, standard deviation (S.D.) and MSE for μ among 100 replications for each configuration using different combination weights comparing with full sample MLE.*

Config.		Equal	Prop	Approx. sc.	Scalar	Approx. opt.	ML
	Mean	-1.72277E-02	-1.51605E-02	-1.51605E-02	-1.21296E-02	-1.21296E-02	-1.56028E-02
1	S.D.	(1.32751E-01)	(1.28989E-01)	(1.28989E-01)	(1.32435E-01)	(1.32435E-01)	(1.29150E-01)
	MSE	1.77434E-02	1.67015E-02	1.67015E-02	1.75108E-02	1.75108E-02	1.67564E-02
	Mean	-9.39926E-04	6.93419E-04	6.93419E-04	1.27613E-03	1.27613E-03	7.59533E-04
2	S.D.	(3.93526E-02)	(3.95534E-02)	(3.95534E-02)	(3.84321E-02)	(3.84321E-02)	(3.83996E-02)
	MSE	1.53403E-03	1.54930E-03	1.54930E-03	1.46389E-03	1.46389E-03	1.46036E-03
	Mean	-8.30934E-04	-1.25609E-03	-1.25609E-03	-1.26810E-03	-1.26810E-03	-1.31356E-03
3	S.D.	(1.44839E-02)	(1.47545E-02)	(1.47545E-02)	(1.41310E-02)	(1.41310E-02)	(1.41704E-02)
	MSE	2.08376E-04	2.17095E-04	2.17095E-04	1.99298E-04	1.99298E-04	2.00517E-04
	Mean	9.30928E-03	2.53713E-03	2.53713E-03	8.29367E-03	8.29367E-03	2.82086E-03
4	S.D.	(9.26881E-02)	(8.32009E-02)	(8.32009E-02)	(9.76672E-02)	(9.76672E-02)	(8.28999E-02)
	MSE	8.59183E-03	6.85960E-03	6.85960E-03	9.51227E-03	9.51227E-03	6.81163E-03
	Mean	9.77532E-03	1.02173E-02	1.02173E-02	8.72381E-03	8.72381E-03	1.02422E-02
5	S.D.	(1.09769E-01)	(1.04876E-01)	(1.04876E-01)	(1.04982E-01)	(1.04982E-01)	(1.04847E-01)
	MSE	1.20243E-02	1.09934E-02	1.09934E-02	1.09872E-02	1.09872E-02	1.09880E-02

Based on the simulation results, it appears that using equal weights is not recommended, while using proportional weights produces results com-

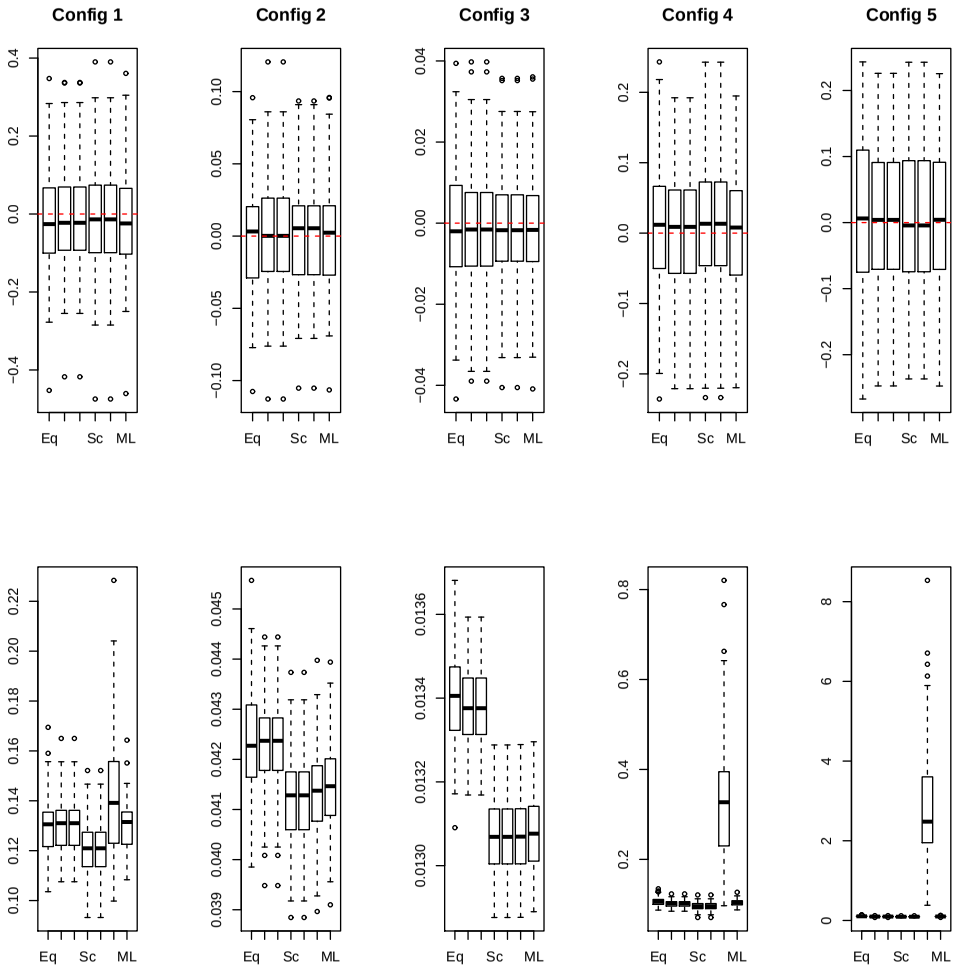

Figure 10: *Second simulation study. Estimates for μ (first row) and its standard error (second row).*

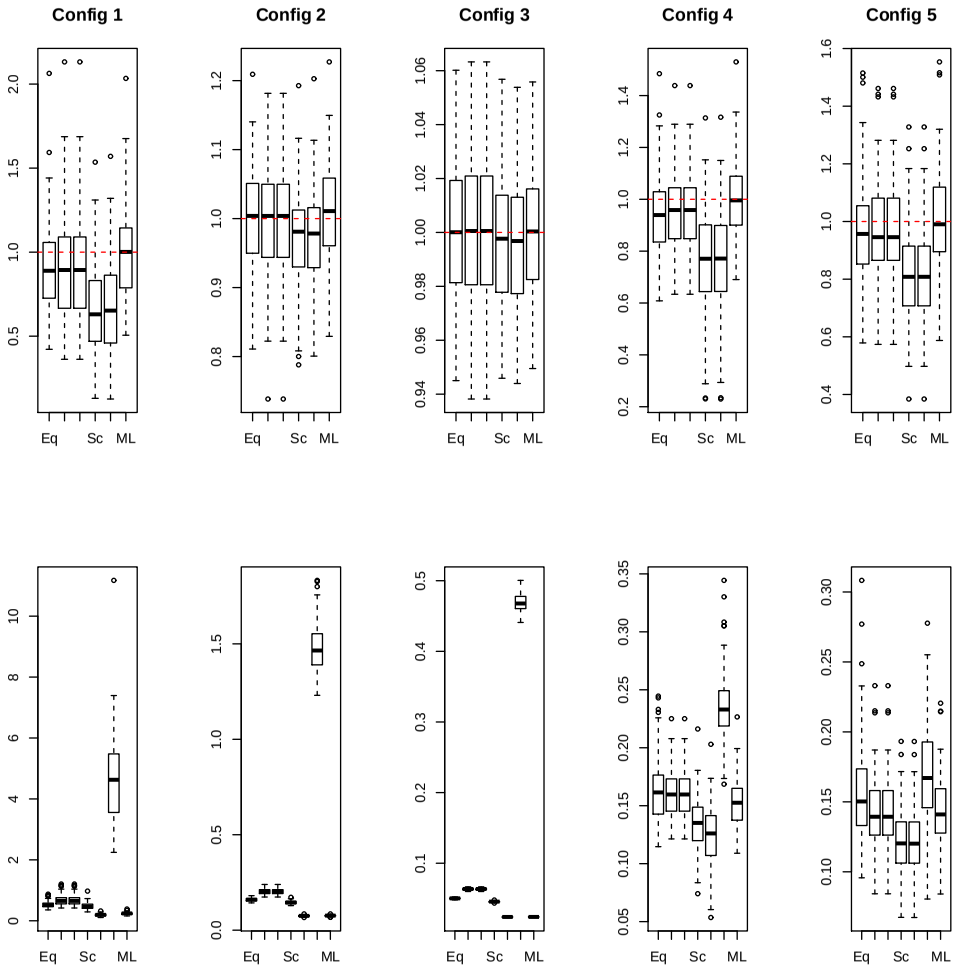

Figure 11: *Second simulation study. Estimates for d (first row) and standard errors (second row).*

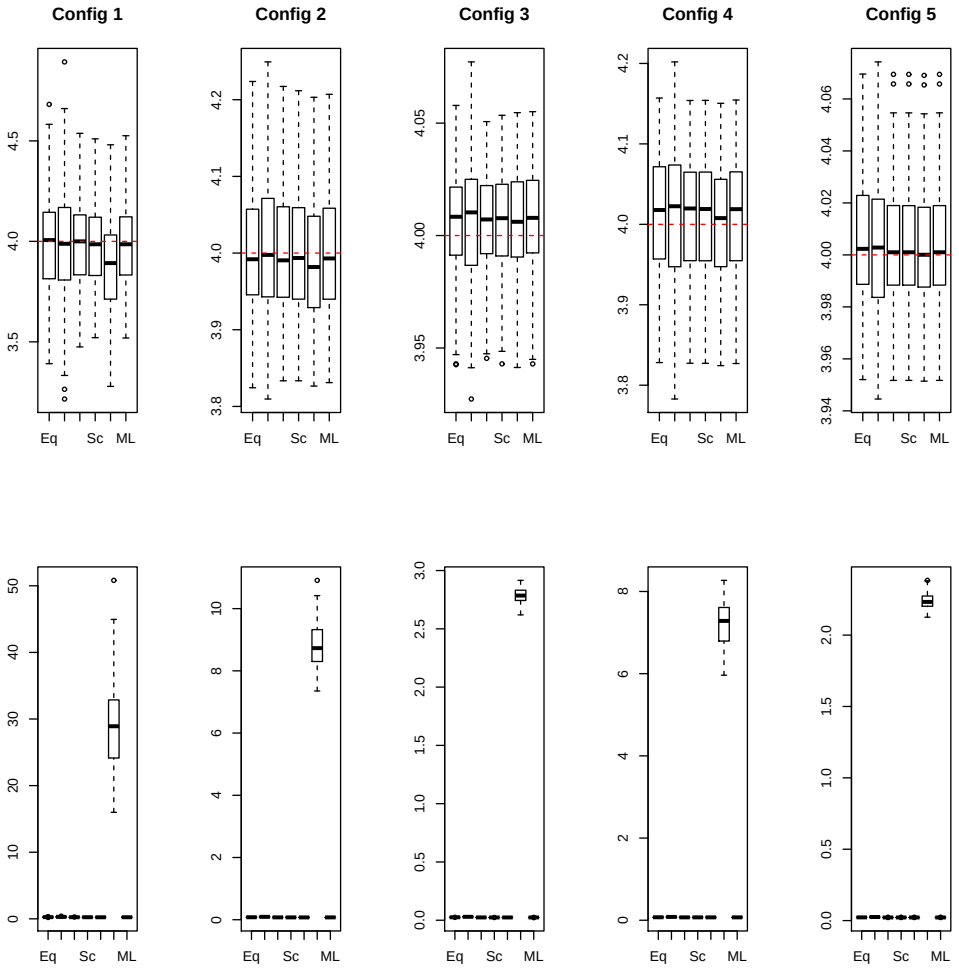

Figure 12: *Second simulation study. Estimates for σ^2 (first row) and standard errors (second row).*

Table 5: *Second simulation study. Mean and standard deviation (S.D.) for standard errors of μ estimates in 100 replications for each configuration using different combination weights comparing with full sample MLE.*

Config.		Equal	Prop	Approx. sc.	Scalar	Simple opt.	Proper opt.	ML
1	Mean	1.29844E-01	1.29733E-01	1.29733E-01	1.20593E-01	1.20593E-01	1.41085E-01	1.29648E-01
	S.D.	(1.18175E-02)	(1.07496E-02)	(1.07496E-02)	(1.09122E-02)	(1.09122E-02)	(2.25155E-02)	(9.95614E-03)
2	Mean	4.23655E-02	4.23056E-02	4.23056E-02	4.11657E-02	4.11657E-02	4.12635E-02	4.14298E-02
	S.D.	(1.08386E-03)	(8.88747E-04)	(8.88747E-04)	(8.71465E-04)	(8.71465E-04)	(8.90259E-04)	(8.66432E-04)
3	Mean	1.34008E-02	1.33822E-02	1.33822E-02	1.30725E-02	1.30725E-02	1.30728E-02	1.30799E-02
	S.D.	(1.21880E-04)	(9.92324E-05)	(9.92324E-05)	(9.68871E-05)	(9.68871E-05)	(9.68974E-05)	(9.62652E-05)
4	Mean	1.06373E-01	1.01232E-01	1.01232E-01	9.60807E-02	9.60807E-02	3.30358E-01	1.03382E-01
	S.D.	(9.80278E-03)	(7.26031E-03)	(7.26031E-03)	(8.36242E-03)	(8.36242E-03)	(1.39042E-01)	(7.17708E-03)
4	Mean	1.05176E-01	9.84615E-02	9.84615E-02	9.45066E-02	9.45066E-02	2.81427E+00	1.00533E-01
	S.D.	(1.10711E-02)	(8.18259E-03)	(8.18259E-03)	(8.14229E-03)	(8.14229E-03)	(1.41211E+00)	(8.28537E-03)

parable with ML. Of course, in case of σ^2 the approximate scalar weights work better comparing with ML. An interesting outcome of the simulation is that by keeping the number of clusters of different sizes constant, but allowing the cluster sizes to increase, improves estimation of σ^2, while increasing the number of clusters and keeping their sizes constant improves the estimation of d. This is not surprising, because d is the between-cluster variability, which is easier to estimate from a larger number of clusters. This should be seen against the background of relatively small differences anyway.

The results based on MI are not comparable with sample-splitting results. In particular, the variance component d is underestimated using MI, while σ^2 is overestimated. The larger standard errors in this case suggest that the sample-splitting methods use information more efficiently.

Comparing computation times, the closed-form approaches are the clear

Table 6: *Second simulation study. Mean, standard deviation (S.D.) and MSE for d estimates in 100 replications for each configuration using different combination weights comparing with full sample MLE.*

Config.		Equal	Prop	Approx. sc.	Scalar	Approx. opt	ML
	Mean	9.09580E-01	9.00788E-01	9.00788E-01	6.63293E-01	6.71111E-01	9.86549E-01
1	S.D.	(2.66885E-01)	(2.99736E-01)	(2.99736E-01)	(2.70556E-01)	(2.79060E-01)	(2.55874E-01)
	MSE	7.86910E-02	9.87862E-02	9.87862E-02	1.85840E-01	1.85264E-01	6.49975E-02
	Mean	9.98984E-01	9.97534E-01	9.97534E-01	9.72269E-01	9.73771E-01	1.00712E+00
2	S.D.	(7.44958E-02)	(8.28143E-02)	(8.28143E-02)	(7.28186E-02)	(7.22762E-02)	(7.08463E-02)
	MSE	5.49515E-03	6.79570E-03	6.79570E-03	6.01854E-03	5.85958E-03	5.01969E-03
	Mean	1.00004E+00	9.99697E-01	9.99697E-01	9.97218E-01	9.97153E-01	1.00019E+00
3	S.D.	(2.61046E-02)	(2.82480E-02)	(2.82480E-02)	(2.48989E-02)	(2.48068E-02)	(2.44715E-02)
	MSE	6.74637E-04	7.90063E-04	7.90063E-04	6.21496E-04	6.17332E-04	5.92900E-04
	Mean	9.40257E-01	9.50756E-01	9.50756E-01	7.65219E-01	7.65362E-01	9.96005E-01
4	S.D.	(1.53414E-01)	(1.48216E-01)	(1.48216E-01)	(1.91865E-01)	(1.91614E-01)	(1.49451E-01)
	MSE	2.68697E-02	2.41734E-02	2.41734E-02	9.15661E-02	9.14038E-02	2.21282E-02
	Mean	9.63459E-01	9.68182E-01	9.68182E-01	8.21266E-01	8.21270E-01	1.00958E+00
5	S.D.	(1.73767E-01)	(1.63471E-01)	(1.63471E-01)	(1.72162E-01)	(1.72163E-01)	(1.69235E-01)
	MSE	3.12283E-02	2.74677E-02	2.74677E-02	6.12894E-02	6.12881E-02	2.84459E-02

winners, further enhanced by smaller standard errors. Furthermore, it follows that computing the estimates in a semi-parallel fashion, thus avoiding 'for' loops within the splits but using then between splits, is most efficient. Unless the n_k's are very large, computing all estimates at once is the most efficient. Of course, if the estimates for different splits can be done in parallel (without 'for' loops), this is more efficient than estimating them all at once.

S8 Analysis of the NTP Data Using R

First, this Appendix briefly describes the use of some building blocks in R for the analysis of the case study discussed in Section 8. The text file

Table 7: *Second simulation study. Mean and standard deviation (S.D.) for standard errors of d estimates in 100 replications for each configuration using different combination weights comparing with full sample MLE.*

Config.		Equal	Prop.	Approx. sc.	Scalar	Simple opt.	Proper opt.	ML
1	Mean	5.30888E-01	6.76159E-01	6.76159E-01	4.83519E-01	1.95328E-01	4.66561E+00	2.36408E-01
	S.D.	(1.00495E-01)	(1.66365E-01)	(1.66365E-01)	(1.10589E-01)	(4.01066E-02)	(1.40173E+00)	(3.91752E-02)
2	Mean	1.60365E-01	2.03126E-01	2.03126E-01	1.45284E-01	7.48056E-02	1.47955E+00	7.60798E-02
	S.D.	(8.29444E-03)	(1.42616E-02)	(1.42616E-02)	(8.70034E-03)	(3.35411E-03)	(1.31587E-01)	(3.26483E-03)
3	Mean	5.02596E-02	6.34861E-02	6.34861E-02	4.56061E-02	2.39463E-02	4.67989E-01	2.39748E-02
	S.D.	(8.43976E-04)	(1.44201E-03)	(1.44201E-03)	(8.34510E-04)	(3.68251E-04)	(1.24414E-02)	(3.66865E-04)
4	Mean	1.62410E-01	1.59656E-01	1.59656E-01	1.34256E-01	1.24552E-01	2.35351E-01	1.51740E-01
	S.D.	(2.83009E-02)	(2.05482E-02)	(2.05482E-02)	(2.31559E-02)	(2.51091E-02)	(3.00109E-02)	(2.11541E-02)
5	Mean	1.54122E-01	1.43313E-01	1.43313E-01	1.22117E-01	1.22024E-01	1.70868E-01	1.43764E-01
	S.D.	(3.48725E-02)	(2.50371E-02)	(2.50371E-02)	(2.30972E-02)	(2.31064E-02)	(3.70951E-02)	(2.38992E-02)

containing these functions is available for download at `www.ibiostat.be`.

- `splitmeth(idvector, yvector)`: computes the size of each cluster, giving the identification vector and the responses. Afterwards it combines the clusters with the same size together in a block.

- `ckblock(idvector, yvector)`: Identification vectors and outcomes isolated per cluster size in the previous function are used in this function seperately. For each block the cluster size, the size of the block, the mean, variance, correlation and the variance-covariance matrix are computed.

- `estimators(idvector, yvector)`: computes all the estimators mentioned in this paper, given the identification vector and the outcomes. It uses the two functions above, `splitmeth` to split the data according to cluster size and `ckblock`to compute the estimators in each block.

Table 8: *Second simulation study. Mean, standard deviation (S.D.) and MSE for σ^2 estimates in 100 replications for each configuration using different combination weights comparing with full sample MLE.*

Config.		Equal	Prop.	Approx. sc.	Scalar	Approx. opt.	ML
	Mean	3.98608E+00	3.99739E+00	3.98571E+00	3.98364E+00	3.87487E+00	3.98075E+00
1	S.D.	(2.50882E-01)	(2.95821E-01)	(2.35138E-01)	(2.33650E-01)	(2.38167E-01)	(2.30477E-01)
	MSE	6.25062E-02	8.66420E-02	5.49414E-02	5.43140E-02	7.18141E-02	5.29588E-02
	Mean	4.00184E+00	4.00681E+00	4.00177E+00	4.00087E+00	3.99027E+00	4.00064E+00
2	S.D.	(7.85190E-02)	(9.05849E-02)	(7.57443E-02)	(7.56486E-02)	(7.50477E-02)	(7.51739E-02)
	MSE	6.10698E-03	8.16998E-03	5.68296E-03	5.66624E-03	5.67055E-03	5.59502E-03
	Mean	4.00509E+00	4.00491E+00	4.00575E+00	4.00590E+00	4.00472E+00	4.00587E+00
3	S.D.	(2.45760E-02)	(2.82395E-02)	(2.36027E-02)	(2.36983E-02)	(2.37694E-02)	(2.36697E-02)
	MSE	6.23828E-04	8.13646E-04	5.84605E-04	5.90806E-04	5.81652E-04	5.89081E-04
	Mean	4.01402E+00	4.01190E+00	4.01302E+00	4.01304E+00	4.00363E+00	4.01306E+00
4	S.D.	(7.69655E-02)	(8.78962E-02)	(7.32088E-02)	(7.31202E-02)	(7.20589E-02)	(7.31207E-02)
	MSE	6.06101E-03	7.79021E-03	5.47558E-03	5.46319E-03	5.15372E-03	5.46370E-03
	Mean	4.00346E+00	4.00338E+00	4.00292E+00	4.00292E+00	4.00192E+00	4.00292E+00
5	S.D.	(2.56561E-02)	(2.79741E-02)	(2.46670E-02)	(2.46664E-02)	(2.46901E-02)	(2.46669E-02)
	MSE	6.63599E-04	7.86128E-04	6.10884E-04	6.10854E-04	6.07187E-04	6.10909E-04

The function gives the estimators of μ, σ^2 and d together with their precision.

Next, the R functions to compute CS sample splitting estimates and variances are described. Section S8.1 will describe the input of the functions, the output of each function will de described in Section S8.2. Each function is followed by an example as well. The functions themselves are presented in Section S8.3.

There are three functions provided to estimate CS parameters (μ, σ^2, d). The function `est.CS` estimates the CS parameters and their variances using vectorized (semi-parallel) calculations, i.e. the for loops are avoided for calculation within each split. The function `est.CS.for` implements the for-

Table 9: *Second simulation study. Mean and standard deviation (S.D.) for standard errors of σ^2 estimates in 100 replications for each configuration using different combination weights comparing with full sample MLE.*

Config.		Equal	Prop.	Approx. sc.	Scalar	Simple opt.	Proper opt.	ML
1	Mean	2.53758E-01	2.95524E-01	2.40482E-01	2.38699E-01	2.30855E-01	2.88730E+01	2.35804E-01
	S.D.	(1.98892E-02)	(3.33124E-02)	(1.45355E-02)	(1.39849E-02)	(1.38813E-02)	(6.93951E+00)	(1.34905E-02)
2	Mean	7.98210E-02	9.26570E-02	7.57927E-02	7.53242E-02	7.48395E-02	8.82494E+00	7.49872E-02
	S.D.	(1.84469E-03)	(3.08841E-03)	(1.45443E-03)	(1.42354E-03)	(1.40575E-03)	(6.92140E-01)	(1.39989E-03)
3	Mean	2.52202E-02	2.92034E-02	2.39687E-02	2.38353E-02	2.37400E-02	2.78559E+00	2.37444E-02
	S.D.	(1.85701E-04)	(3.12856E-04)	(1.42587E-04)	(1.40888E-04)	(1.40071E-04)	(6.70930E-02)	(1.39784E-04)
4	Mean	7.22378E-02	8.07793E-02	7.01684E-02	7.01663E-02	6.99970E-02	7.23107E+00	7.01238E-02
	S.D.	(1.54384E-03)	(2.35043E-03)	(1.28712E-03)	(1.28385E-03)	(1.26263E-03)	(5.28710E-01)	(1.27765E-03)
5	Mean	2.25782E-02	2.52054E-02	2.19702E-02	2.19702E-02	2.19647E-02	2.23651E+00	2.19689E-02
	S.D.	(1.55999E-04)	(2.29649E-04)	(1.35364E-04)	(1.35358E-04)	(1.35462E-04)	(5.50362E-02)	(1.35379E-04)

mulas in (4.5) directly using for loops. The function `est.CS.all` estimates the parameters for all of the splits simultaneously.

The function `param.free.CS` can be used to combine results from different sub-samples using parameter-free weights: Prop., Equal, and Appr.sc. The function `scalar.weights.CS` gives the same results but using scalar weights (approximated by sub-sample specific estimates). The function `approx.optimal.CS` combines the results of different sub-samples using approximated optimal weights, the proper variances for these estimates are also provided. Finally, the function `clusterBYcluster.CS` computes the cluster specific estimate of (μ, σ^2, d) using cluster-by-cluster approaches. Combining them by the desired rule using the three previous functions is straightforward.

Table 10: *Second simulation study. Computation time (in seconds) using closed-form solutions with different implementation forms, compared to PROC MIXED.*

Config.		Split by split without loops	Split by split using loops	Together	PROC MIXED
1	Mean	**0.00520**	0.00980	0.00640	0.34155
	S.D.	(0.00882)	(0.00943)	(0.00823)	(0.17711)
2	Mean	**0.02150**	0.07340	0.05660	0.35575
	S.D.	(0.01617)	(0.02006)	(0.09662)	(0.03073)
3	Mean	**0.17980**	0.73480	0.43400	0.81783
	S.D.	(0.02292)	(0.05835)	(0.11543)	(0.02582)
4	Mean	**0.00220**	0.00610	0.00360	2.58808
	S.D.	(0.00579)	(0.01497)	(0.00689)	(0.40720)
5	Mean	0.04030	0.27490	**0.02130**	629.58333
	S.D.	(0.01521)	(0.01941)	(0.00872)	(116.09435)

Table 11: *Second simulation study. Mean, standard deviation (S.D.) and MSE for CS parameter estimates in 100 replications for configuration 2 using different combination weights comparing with full sample MLE and MI-MLE.*

		Equal	Prop.	Approx. sc.	Scalar	Approx. opt.	MI	ML
μ	Mean	-5.09643E-03	-2.62061E-03	-2.62061E-03	-2.88933E-03	-2.88933E-03	-8.04195E-03	-3.28224E-03
	S.D.	(4.85424E-02)	(4.91772E-02)	(4.91772E-02)	(4.68032E-02)	(4.68032E-02)	(6.00662E-02)	(4.71723E-02)
	MSE	2.35877E-03	2.40108E-03	2.40108E-03	2.17698E-03	2.17698E-03	3.63654E-03	2.21375E-03
d	Mean	9.98392E-01	9.95216E-01	9.95216E-01	9.70589E-01	9.71627E-01	3.51123E-02	9.92960E-01
	S.D.	(7.33193E-02	(7.46946E-02)	(7.46946E-02)	(7.59516E-02)	(7.53362E-02)	(1.83983E-02)	(7.50610E-02)
	MSE	5.32455E-03	5.54638E-03	5.54638E-03	6.57598E-03	6.42380E-03	9.31343E-01	5.62737E-03
σ^2	Mean	4.00782E+00	4.00791E+00	4.00581E+00	4.00544E+00	3.99400E+00	5.32627E+00	4.00347E+00
	S.D.	(7.66500E-02)	(8.98881E-02)	(7.19708E-02)	(7.13441E-02)	(7.19080E-02)	(1.55770E-01)	(7.56968E-02)
	MSE	5.87763E-03	8.06169E-03	5.16181E-03	5.06871E-03	5.15505E-03	1.78301E+00	5.68472E-03

S8.1 Input

Table 13 describes the input for the various functions.

S8.2 Output

The output of each function is a list (except for `scalar.weights.CS`) containing the calculated quantities. The functions `est.CS` and `est.CS.for`

Table 12: *Second simulation study. Mean and standard deviation (S.D.) for the standard error of CS parameter estimates in 100 replications for configuration 2 using different combination weights comparing with full sample MLE and MI-MLE.*

		Equal	Prop.	Approx. sc.	Scalar	Simple opt.	Proper opt.	MI	ML
μ	Mean	4.24162E-02	4.22766E-02	4.22766E-02	4.11356E-02	4.11356E-02	4.12439E-02	4.95431E-02	4.14004E-02
	S.D.	(1.21548E-03)	(8.59430E-04)	(8.59430E-04)	(9.31222E-04)	(9.31222E-04)	(9.42598E-04)	(7.65032E-03)	(9.11785E-04)
d	Mean	1.60545E-01	2.02922E-01	2.02922E-01	1.45623E-01	7.47531E-02	1.48865E+00	9.10211E-02	7.54391E-02
	S.D.	(8.88524E-03)	(1.47416E-02)	(1.47416E-02)	(9.84170E-03)	(3.76197E-03)	(1.29914E-01)	(5.18621E-03)	(3.38044E-03)
σ^2	Mean	7.99442E-02	9.26315E-02	7.58626E-02	7.54139E-02	7.49171E-02	8.86450E+00	1.64310E-01	7.50377E-02
	S.D.	(1.87358E-03)	(3.14842E-03)	(1.39531E-03)	(1.34244E-03)	(1.33807E-03)	(6.56714E-01)	(2.27104E-02)	(1.42688E-03)

compute the parameters estimates with their variances within each split. Table 14 presents the output for these two functions. An example of using them is given in Example 1. For estimating the parameters for all of the splits simultaneously, function `est.CS.all` can be used. The output description of this function can be found in Table 15. An example of using it is presented in Example 2. One may denote that the input dataset for using this function should make all clusters of equal size using missing values `NaN`. The function `param.free.CS` computes the three parameter free combining rules: Prop., Equal, and Appr.sc., Table 16 presents the output of this function, an example of using it is given in Example 3. For computing the scalar weights one may use the function `scalar.weights.CS`. The output of this function are described in Table 17 and Example 4 shows how to use it in practice. Function `approx.optimal.CS` computes the approximate optimal weights together with their proper variances. Table 18 gives the output of this function and Example 5 will show the use of this

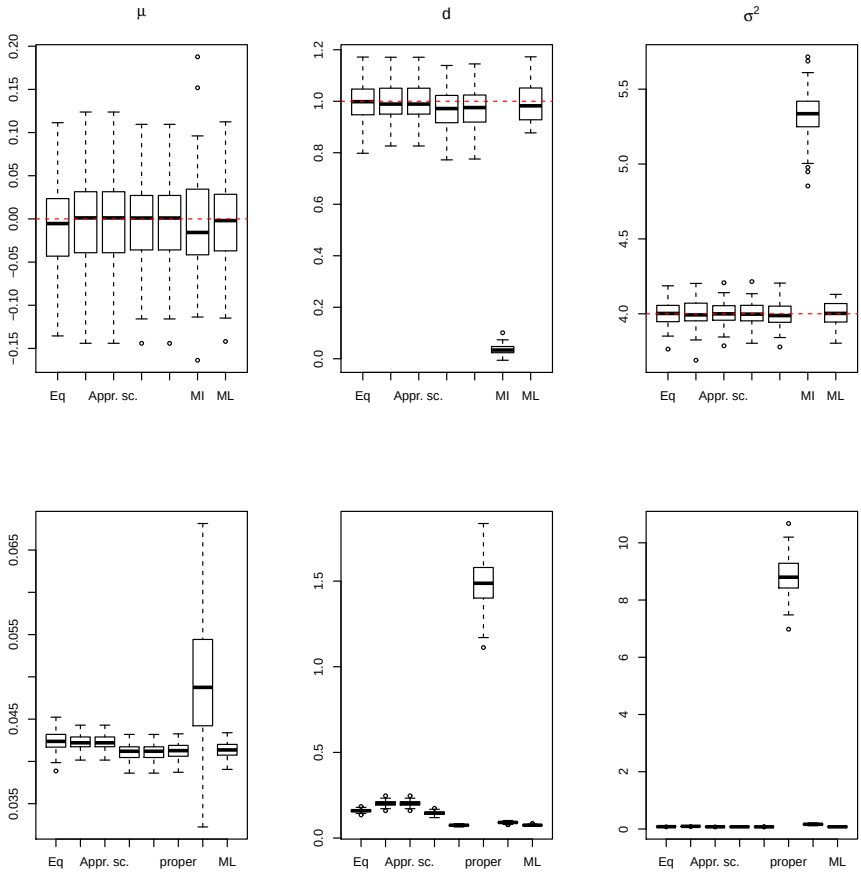

Figure 13: *Second simulation study. Estimated CS parameters (first row) and their standard error (second row) using sample splitting, MI-MLE, and MLE.*

function. Finally, for cluster-by-cluster analyses, the three methods discussed (weighted, two stage, unbiased two stage) are implemented in the function `clusterBYcluster.CS`. One may find descriptions of the output of this function together with an example of using it in Table 19 and Example 6, respectively.

Table 13: *Function input.*

Input	Description
`n`	an integer indicating the common cluster size within each split
`C`	an integer indicating the number of clusters within each split
`Y`	a vector containing the response values corresponding to each split
`data`	a 2-column matrix with response variable as its first column and splits indicator as its second column. All clusters should be of equal size by placing `NaN` to make smaller clusters the same size as the largest one.
`nk`	a vector containing the cluster sizes
`ck`	a vector containing the number of clusters of size n_k
`mu.split.est`	a vector containing the $\widehat{\mu}$'s from all sub-samples.
`sigma2.split.est`	a vector containing the $\widehat{\sigma}^2$'s from all sub-samples.
`d.split.est`	a vector containing the $\widehat{d}$'s from all sub-samples.
`mu.split.var`	a vector containing the $\mathrm{Var}(\widehat{\mu})$'s from all sub-samples.
`sigma2.split.var`	a vector containing the $\mathrm{Var}(\widehat{\sigma}^2)$'s from all sub-samples.
`d.split.var`	a vector containing the $\mathrm{Var}(\widehat{d})$'s from all sub-samples.
`Data`	(only applicable in function `clusterBYcluster.CS`) a 3-column matrix with first column the subject, second column the response variable, and third column the split indexes, which show which observation belongs to which sub-sample.

Example 1.

```
> est.CS(n,C,Y)

$mu.hat

[1] -0.02008066



$d.hat

[1] 1.05208
```

Table 14: *Function output:* `est.CS` *and* `est.CS.for`

Output	Description
`mu.hat`	a scalar presenting $\widehat{\mu}$.
`d.hat`	a scalar presenting $\widehat{d}$.
`sigma2.hat`	a scalar presenting $\widehat{\sigma^2}$.
`var.mu.hat`	a scalar presenting variance of $\widehat{\mu}$.
`cov.varcomp`	a matrix presenting covariance matrix of $(\widehat{\sigma^2}, \widehat{d})$

```
$sigma2.hat

[1] 3.893208


$var.mu.hat

[1] 0.01025821


$cov.varcomp

             [,1]         [,2]

[1,]  0.028870611 -0.003608826

[2,] -0.003608826  0.032020343
```

Example 2.

```
> est.CS.all(data2,ck,nk)

$mu.hat
```

Table 15: *Function output:* `est.CS.all`

Output	Description
`mu.hat`	a vector presenting $\widehat{\mu}$ for each split.
`d.hat`	a vector presenting $\widehat{d}$ for each split.
`sigma2.hat`	a vector presenting $\widehat{\sigma^2}$ for each split.
`var.mu.hat`	a vector presenting variance of $\widehat{\mu}$ for each split.
`var.d.hat`	a vector presenting variance of $\widehat{d}$ for each split.
`var.sigma2.hat`	a vector presenting variance of $\widehat{\sigma^2}$ for each split.
`cov.d.sigma2.hat`	a vector presenting covariance of $(\widehat{\sigma^2}, \widehat{d})$ for each split.

```
             [,1]

[1,]  0.017154093

[2,]  0.043479563

[3,] -0.156958273

[4,] -0.031153966

[5,] -0.007420771


$sigma2.hat

[1] 3.893208 4.085951 3.901042 4.015487 3.900566


$d.hat

[1] 1.0506935 1.0162879 0.7930084 0.8279435 0.9344471


$var.mu.hat
```

Table 16: *Function output:* `param.free.CS`

Output	Description
`mu`	a matrix with columns $\tilde{\mu}$ and $\mathrm{Var}(\tilde{\mu})$ for Equal, Proper, and Appr.sc. weights, rows 1–3, respectively.
`sigma2`	a matrix with columns $\tilde{\sigma}^2$ and $\mathrm{Var}(\tilde{\sigma}^2)$ for Equal, Proper, and Appr.sc. weights, rows 1–3, respectively.
`d`	a matrix with columns $\tilde{d}$ and $\mathrm{Var}(\tilde{d})$ for Equal, Proper, and Appr.sc. weights, rows 1–3, respectively.

```
[1] 0.010248964 0.007333913 0.006977852 0.006370544 0.011944848


$var.d.hat

[1] 0.03196348 0.02822874 0.03485060 0.01648236 0.02863248


$var.sigma2.hat

[1] 0.02887061 0.03339000 0.05072710 0.02015517 0.02173487


$cov.d.sigma2.hat

[1] -0.003608826 -0.006678000 -0.016909033 -0.002239464

-0.001448992
```

Example 3.

```
> param.free.CS (nk,ck,mu.split.est,sigma2.split.est,
```

```
                d.split.est,mu.split.var,sigma2.split.var,
                d.split.var)
$mu
               Est          Var
Equal    0.8486860 0.0001863644
Prop     0.8350032 0.0001755510
Appr.sc. 0.8350032 0.0001755510

$sigma2
                 Est          Var
Equal    0.008473228 2.368028e-07
Prop     0.008471484 1.973433e-07
Appr.sc. 0.008327288 1.725952e-07

$d
                Est          Var
Equal    0.01443741 5.351632e-06
Prop     0.01538224 5.861071e-06
Appr.sc. 0.01538224 5.861071e-06
```

Table 17: *Function output:* `scalar.weights.CS`

Output	Description
-	a matrix with columns $\tilde{\mu}$ and $\mathrm{Var}(\tilde{\mu})$ for scalar weights.

Table 18: *Function output:* `approx.optimal.CS`

Output	Description
`mu.est`	$\tilde{\mu}$ obtained by approximate optimal weights
`varcomp.est`	$(\tilde{\sigma}^2, \tilde{d})$ obtained by approximated optimal weights
`mu.var`	$\mathrm{Var}(\tilde{\mu})$ obtained by approximate optimal weights
`varcomp.var`	covariance matrix of $(\tilde{\sigma}^2, \tilde{d})$ obtained by approximated optimal weights
`proper.var.mu`	$\mathrm{Var}_{\mathrm{Proper}}(\tilde{\mu})$ obtained by approximate optimal weights
`proper.var.varcomp`	proper covariance matrix of $(\tilde{\sigma}^2, \tilde{d})$ obtained by approximate optimal weights

Example 4.

```
scalar.weights.CS (nk,ck,mu.split.est,sigma2.split.est,

                   d.split.est,mu.split.var,sigma2.split.var,

                   d.split.var)

              Est.          Var

mu     0.841588794 1.700247e-04

sigma2 0.008313578 1.716135e-07

d      0.013715408 4.986101e-06
```

Example 5.

```
approx.optimal.CS (nk,ck,sigma2.split.est,d.split.est,
                   sigma2.split.var,d.split.var)
$mu.est
[1] 0.8415888


$varcomp.est
[1] 0.006059707 0.014069710


$mu.var
[1] 0.0001771469


$varcomp.var
              [,1]          [,2]
[1,]  7.503554e-08 -1.089792e-08
[2,] -1.089792e-08  5.144086e-06


$proper.var.mu
[1] 0.0001771471


$proper.var.varcomp
```

Table 19: *Function output:* `clusterBYcluster.CS`

Output	Description
`mu.split.est`	a vector containing cluster-by-cluster estimated μ for each cluster
`mu.split.var`	a matrix with rows the variances of estimated μ for each cluster and with columns indicating the three methods: weighted, two-stage, and unbiased two-stage
`sigma2.split.est`	a matrix with rows the estimated σ^2 for each cluster and columns indicating the three methods: weighted ($\widehat{\sigma}^2$), two-stage (s^2) and unbiased two-stage (s_*^2)
`sigma2.split.var`	a matrix with rows the variances of estimated σ^2 for each cluster and its columns indicate the three methods: weighted ($\widehat{\sigma}^2$), two stage (s^2) and unbiased two stage (s_*^2)
`d.split.est`	a matrix with rows the estimated d^2 for each cluster and columns indicating the three methods: weighted ($\widehat{\sigma}^2$), two-stage (t^2), and unbiased two-stage (t_*^2)
`d.split.var`	a matrix with rows the variances of estimated d^2 for each cluster and columns indicating the three methods: weighted ($\widehat{\sigma}^2$), two-stage (t^2) and unbiased two-stage (t_*^2)
`Var.varcomp`	a list containing matrices as its elements. The matrices are the full covariance matrices for $(\widehat{\sigma}^2, \widehat{d})$ from a weighted cluster-by-cluster analysis

```
              [,1]          [,2]

[1,]  8.045528e-06 -9.606275e-06

[2,] -9.606275e-06  2.111345e-05
```

One may note that once the cluster-specific estimates have been obtained using any of the cluster-by-cluster approaches, one can easily combine them with the desired rule using the available functions. The output

are organized in accordance with the input of other functions, so they can be directly plugged in.

Example 6.

```
> clusterBYcluster.CS(nk,ck,Data)

$mu.split.est

[1] 0.8058733 0.8500635 0.8350303 0.8645111 0.9579167 0.7938929

0.8111373  0.8538187


$mu.split.var

        Weighted    TwoStage TwoStageUnbiased

[1,] 0.001307539 0.001376916      0.001478020

[2,] 0.001650106 0.001810324      0.002043291

[3,] 0.001334065 0.001390168      0.001517057

[4,] 0.001092201 0.001189274      0.001332732

[5,] 0.003312091 0.003506955      0.005190838

[6,] 0.003285979 0.003373701      0.003849875

[7,] 0.001098199 0.001127220      0.001198495

[8,] 0.001034920 0.001055199      0.001136498


$sigma2.split.est
```

```
        Weighted    TwoStage TwoStageUnbiased
[1,] 0.011562832 0.010406549      0.011562832
[2,] 0.011776069 0.010093773      0.011776069
[3,] 0.008146211 0.007405646      0.008146211
[4,] 0.014040871 0.013104813      0.014040871
[5,] 0.005344821 0.004676719      0.005344821
[6,] 0.010580691 0.009824927      0.010580691
[7,] 0.006458361 0.005920164      0.006458361
[8,] 0.003998299 0.003690738      0.003998299


$sigma2.split.var
         Weighted     TwoStage TwoStageUnbiased
[1,] 1.980727e-06 1.299555e-06     1.980727e-06
[2,] 5.136141e-06 2.772361e-06     5.136141e-06
[3,] 1.106012e-06 7.554214e-07     1.106012e-06
[4,] 3.129302e-06 2.374623e-06     3.129302e-06
[5,] 2.720678e-06 1.594811e-06     2.720678e-06
[6,] 2.152904e-06 1.600612e-06     2.152904e-06
[7,] 4.461008e-07 3.149770e-07     4.461008e-07
[8,] 1.903143e-07 1.381729e-07     1.903143e-07
```

```
$d.split.est

        Weighted    TwoStage TwoStageUnbiased

[1,] 0.018456799 0.019613082       0.02101402

[2,] 0.013168657 0.014850953       0.01670732

[3,] 0.015268213 0.016008777       0.01746412

[4,] 0.008893749 0.009829807       0.01105853

[5,] 0.009268172 0.009936274       0.01490441

[6,] 0.025532065 0.026287829       0.03004323

[7,] 0.018131189 0.018669386       0.01983622

[8,] 0.014181322 0.014488883       0.01560341


$d.split.var

         Weighted    TwoStage TwoStageUnbiased

[1,] 5.130954e-05 0.007715211     6.553629e-05

[2,] 4.911611e-05 0.008261736     7.515067e-05

[3,] 4.272263e-05 0.005140078     5.523509e-05

[4,] 2.148616e-05 0.003950645     3.197116e-05

[5,] 6.586221e-05 0.002231365     1.616688e-04

[6,] 1.727735e-04 0.006083359     2.371446e-04
```

```
[7,] 4.100850e-05 0.005122242     4.883729e-05

[8,] 2.999080e-05 0.003263327     3.616558e-05


$Var.varcomp

$Var.varcomp[[1]]

              [,1]          [,2]

[1,]  1.980727e-06 -1.980727e-07

[2,] -1.980727e-07  5.130954e-05


$Var.varcomp[[2]]

              [,1]          [,2]

[1,]  5.136141e-06 -7.337344e-07

[2,] -7.337344e-07  4.911611e-05


$Var.varcomp[[3]]

              [,1]          [,2]

[1,]  1.106012e-06 -1.005466e-07

[2,] -1.005466e-07  4.272263e-05


$Var.varcomp[[4]]
```

```
              [,1]          [,2]
[1,]  3.129302e-06 -2.086202e-07
[2,] -2.086202e-07  2.148616e-05


$Var.varcomp[[5]]
              [,1]          [,2]
[1,]  2.720678e-06 -3.400847e-07
[2,] -3.400847e-07  6.586221e-05


$Var.varcomp[[6]]
              [,1]          [,2]
[1,]  2.152904e-06 -1.537789e-07
[2,] -1.537789e-07  1.727735e-04


$Var.varcomp[[7]]
              [,1]          [,2]
[1,]  4.461008e-07 -3.717507e-08
[2,] -3.717507e-08  4.100850e-05


$Var.varcomp[[8]]
```

```
              [,1]          [,2]
[1,]  1.903143e-07 -1.463956e-08
[2,] -1.463956e-08  2.999080e-05
```

S8.3 R Functions

Here are the R functions.

```
est.CS <- function(n,C,Y){
y.matrix=matrix(Y,n,C)
mu.hat=mean(Y)
Z=Y-mu.hat
Z.matrix=matrix(Z,n,C)
J=matrix(1,n,n)
tmp1=sum(apply(Z.matrix^2,2,sum))
tmp2=apply(Z.matrix%*%matrix(apply(Z.matrix,2,sum),C,1),2,sum)
tmp3=1/((C*n)*(n-1))
sigma2.hat=tmp3*((n*sum(tmp1))-sum(tmp2))
d.hat=tmp3*(sum(tmp2)-sum(tmp1))
var.mu.hat=(sigma2.hat+(n*d.hat))/(C*n)
```

```
cov.varcomp=(2*(sigma2.hat^2)/((C*n)*(n-1)))*matrix(
c(n,-1,-1,(((sigma2.hat^2) + ((2*(n-1))*(d.hat*sigma2.hat)) +
((n*(n-1))*(d.hat^2)))/(sigma2.hat^2))),2,2)
return(list(mu.hat=mu.hat,d.hat=d.hat,sigma2.hat=sigma2.hat
,var.mu.hat=var.mu.hat,cov.varcomp=cov.varcomp))
}


est.CS.for <- function(n,C,Y){
y.matrix=matrix(Y,n,C)
mu.hat=mean(Y)
Z=Y-mu.hat
Z.matrix=matrix(Z,n,C)
tmp2=rep(0,C)
tmp1=rep(0,C)
J=matrix(1,n,n)
for (i in 1:C){
tmp1[i]=t(Z.matrix[,i])%*% Z.matrix[,i]
tmp2[i]= (t(Z.matrix[,i])%*%J)%*%Z.matrix[,i]
}
tmp3=1/((C*n)*(n-1))
```

```
sigma2.hat=tmp3*((n*sum(tmp1))-sum(tmp2))

d.hat=tmp3*(sum(tmp2)-sum(tmp1))

var.mu.hat=(sigma2.hat+(n*d.hat))/(C*n)

cov.varcomp=(2*(sigma2.hat^2)/((C*n)*(n-1)))*matrix(

c(n,-1,-1,(((sigma2.hat^2) + ((2*(n-1))*(d.hat*sigma2.hat)) +

((n*(n-1))*(d.hat^2)))/(sigma2.hat^2))),2,2)

return(list(mu.hat=mu.hat,d.hat=d.hat,sigma2.hat=sigma2.hat,

var.mu.hat=var.mu.hat,cov.varcomp=cov.varcomp))

}


est.CS.all <- function(data,ck,nk){


Y.mat=matrix(data[,1],max(nk),dim(data)[1]/max(nk))

split.idx.sub=matrix(data[,2],max(nk),dim(data)[1]/max(nk))[1,]

split.matrix=matrix(0,length(ck),sum(ck))

for (i in 1:length(ck)){

split.matrix[i,split.idx.sub==i]=1

}

subj.mean=apply(Y.mat,2,sum,na.rm=T)

split.mu.hat=(split.matrix%*%subj.mean)/(ck*nk)
```

```
subj.mu.hat=t(split.matrix)%*%split.mu.hat

mean.mat=matrix(rep(c(subj.mu.hat),max(nk)),max(nk),

                length(subj.mean),byrow=T)

Z.mat=(Y.mat-mean.mat)

tmp1.1=matrix(rep(apply(Z.mat^2,2,sum,na.rm=T),length(ck)),

              length(ck),sum(ck),byrow=T)

tmp1=apply(split.matrix*tmp1.1,1,sum)

tmp2.1=matrix(rep(apply(Z.mat,2,sum,na.rm=T),length(ck)),

              length(ck),sum(ck),byrow=T)

tmp2.2=split.matrix*tmp2.1

Z.mat.zero=Z.mat

Z.mat.zero[is.na(Z.mat)==TRUE]=0

tmp2=apply(Z.mat.zero%*%t(tmp2.2),2,sum,na.rm=T)

tmp3=1/((ck*nk)*(nk-1))

split.sigma2.hat=tmp3* ((nk*tmp1)-tmp2)

split.d.hat=tmp3*(tmp2-tmp1)

split.var.mu.hat=(split.sigma2.hat + (nk*split.d.hat))/(ck*nk)

varcomp.factor=(2*(split.sigma2.hat^2)/((ck*nk)*(nk-1)))

split.var.d.hat=varcomp.factor*(((split.sigma2.hat^2) +

 ((2*(nk-1))*(split.d.hat*split.sigma2.hat)) +
```

```
((nk*(nk-1))*(split.d.hat^2)))/(split.sigma2.hat^2))

split.var.sigma2.hat=varcomp.factor*nk

split.cov.d.sigma2.hat=-1*varcomp.factor

return(list(mu.hat=split.mu.hat,sigma2.hat=split.sigma2.hat,

            d.hat=split.d.hat,var.mu.hat=split.var.mu.hat,

            var.d.hat=split.var.d.hat,

var.sigma2.hat=split.var.sigma2.hat,

cov.d.sigma2.hat=split.cov.d.sigma2.hat))

}


param.free.CS <- function(nk,ck,mu.split.est,sigma2.split.est,

                          d.split.est,mu.split.var,sigma2.split.var,

                          d.split.var){

num.split=length(ck)

# Calculating parameter free weights

Equal=rep(1/num.split,num.split)

Prop=ck/sum(ck)

Appr.sc=(ck*nk)/sum(ck*nk)

# mu

mu.eq=c(sum(Equal*mu.split.est),sum((Equal^2)*mu.split.var))
```

```
mu.prop=c(sum(Prop*mu.split.est),sum((Prop^2)*mu.split.var))

mu.appr.sc=mu.prop

mu=rbind(mu.eq,mu.prop,mu.appr.sc)

colnames(mu)=c("Est","Var")

rownames(mu)=c("Equal","Prop","Appr.sc.")

# Sigma2

sigma2.eq=c(sum(Equal*sigma2.split.est),

            sum((Equal^2)*sigma2.split.var))

sigma2.prop=c(sum(Prop*sigma2.split.est),

              sum((Prop^2)*sigma2.split.var))

sigma2.appr.sc=c(sum(Appr.sc*sigma2.split.est),

                 sum((Appr.sc^2)*sigma2.split.var))

sigma2=rbind(sigma2.eq,sigma2.prop,sigma2.appr.sc)

colnames(sigma2)=c("Est","Var")

rownames(sigma2)=c("Equal","Prop","Appr.sc.")

# d

d.eq=c(sum(Equal*d.split.est),sum((Equal^2)*d.split.var))

d.prop=c(sum(Prop*d.split.est),sum((Prop^2)*d.split.var))

d.appr.sc=d.prop

d=rbind(d.eq,d.prop,d.appr.sc)
```

```
colnames(d)=c("Est","Var")

rownames(d)=c("Equal","Prop","Appr.sc.")


return(list(mu=mu,sigma2=sigma2,d=d))

}


scalar.weights.CS

   <- function(nk,ck,mu.split.est,sigma2.split.est,d.split.est,

               mu.split.var,sigma2.split.var,d.split.var){


ak=(ck*nk)/(sigma2.split.est + (nk*d.split.est))

w.mu=ak/sum(ak)

bk=ck*(nk-1)

w.sigma2=bk/sum(bk)

gk=(ck*nk)/

(((sigma2.split.est^2)/(nk-1))

+((2*sigma2.split.est)*d.split.est)+(nk*(d.split.est^2)))

w.d=gk/sum(gk)


mu.scalar=c(sum(w.mu*mu.split.est), sum(mu.split.var* (w.mu^2)))
```

```
d.scalar=c(sum(w.d*d.split.est), sum(d.split.var* (w.d^2)))

sigma2.scalar=c(sum(w.sigma2*sigma2.split.est),

                sum(sigma2.split.var* (w.sigma2^2)))

param.scalar=rbind(mu.scalar,sigma2.scalar,d.scalar)

colnames(param.scalar)=c("Est.","Var")

rownames(param.scalar)=c("mu","sigma2","d")

return(param.scalar)

}


approx.optimal.CS

   <- function(nk,ck,mu.split.est,sigma2.split.est,d.split.est,

               sigma2.split.var,d.split.var){

library(magic)

num.split=length(ck)

#Calculating approximated optimal weights

W=NULL

for (i in 1:num.split){

V1=(2*(sigma2.split.est[i]^2))/(ck[i]*(nk[i]-1))

V2=(-1)*((2*(sigma2.split.est[i]^2))/(ck[i]*nk[i]*(nk[i]-1)))

V3=(2/(ck[i]*nk[i]))*((sigma2.split.est[i]^2
```

```
            /(nk[i]-1))+((2*d.split.est[i])
             *sigma2.split.est[i])+(nk[i]*(d.split.est[i]^2)))
W[[i]]=solve(matrix(c(V1,V2,V2,V3),2,2))
}


V.total=apply(simplify2array(W),c(1,2),sum)
W.inv=solve(V.total)
W.opt=NULL
sigma2.d=rbind(sigma2.split.est,d.split.est)
sigma2.d.est=matrix(0,dim(sigma2.d)[1],dim(sigma2.d)[2])
for (i in 1:num.split){
W.opt=W.inv %*% W[[i]]
sigma2.d.est[,i]=W.opt%*% sigma2.d[,i]
}
varcomp.est=apply(sigma2.d.est,1,sum)
varcomp.var=W.inv


# Calculating proper Variance


A1=((sigma2.split.est^2) + (nk*d.split.est))^3
```

```
A2=(sigma2.split.est*d.split.est)*((2*sigma2.split.est)

    *(nk*d.split.est))

A3=d.split.est*((sigma2.split.est+(nk*d.split.est))^2)

A=A1-A2-A3


dev.w.sigma2=NULL

dev.w.d=NULL

C3.tmp=NULL

C4.tmp=NULL

for (i in 1:num.split){

tmp1=(-1*ck[i]*nk[i])/A1[i]

tmp2=matrix(c(A[i] ,1 ,1, nk[i]),2,2)

dev.w.sigma2[[i]]=tmp1*tmp2

dev.w.d[[i]]=tmp1 * matrix(c(1,nk[i],nk[i],(nk[i]^2)),2,2)


II=as.matrix(diag(2)-(W.inv%*%W[[i]]))

Theta=c(sigma2.split.est[i],d.split.est[i])

Rho=II%*%Theta

C3.tmp[[i]]=adiag(dev.w.d[[i]],dev.w.sigma2[[i]])

C4.tmp[[i]]= kronecker (diag(2),Rho)
```

```
}
C1.tmp=t(kronecker(c(1,1),diag(2)))
C2.tmp=kronecker(diag(2),W.inv)
proper.var=NULL
for (i in 1:num.split){
C=((C1.tmp%*%C2.tmp)%*%C3.tmp[[i]])%*%C4.tmp[[i]]
AA=W.inv%*%(W[[i]])
B=AA+C
VV=diag(c(d.split.var[i],sigma2.split.var[i]))
proper.var[[i]]=(B%*%VV)%*%t(B)
}
Proper.var=apply(simplify2array(proper.var),c(1,2),sum)


# Approximate weights for mu
Am=(ck*nk)/(sigma2.split.est+(nk*d.split.est))


w.mu=Am/sum(Am)
mu.est=sum(w.mu*mu.split.est)
mu.var=1/sum(Am)
# Proper variance for mu
```

```
proper.var.mu1=mu.var

TMP2=rep(0,num.split)

TMP.1=rep(0,num.split)

for (k in 1:num.split){

TMP1=2*ck[k]*(nk[k]^2)

for (m in 1:num.split){

TMP2[m]=Am[m]*((mu.split.est[k]-mu.split.est[m])^2)

}

TMP.1[k]=TMP1*sum(TMP2)

}

TMP.2=sum(Am)^4

proper.var.mu=sum(TMP.1)/TMP.2

Proper.var.mu=(1/sum(Am))+ proper.var.mu


return(list(mu.est=mu.est,varcomp.est=varcomp.est,mu.var=mu.var,

 varcomp.var=varcomp.var,proper.var.mu=Proper.var.mu,

 proper.var.varcomp=Proper.var))

}


clusterBYcluster.CS <- function(nk,ck,Data){
```

```
# Data should be a 3-column matrix with first column the subject,
# second column the response and third column the split indexes
# which show which observation belongs to which sub-sample.
num.split=length(ck)
Var.varcomp=NULL
mu.split.est=rep(0,num.split)
mu.split.var=matrix(0,num.split,3)
d.split.est=matrix(0,num.split,3)
sigma2.split.est=matrix(0,num.split,3)
d.split.var=matrix(0,num.split,3)
sigma2.split.var=matrix(0,num.split,3)

for (k in 1:num.split){

# Making data for each cluster
split.data=Data[Data[,3]==k,]
n=nk[k]
N=ck[k]
# Computing t^2
data.matrix=matrix(split.data[,2],n,N)
```

```
mu.hat=sum(apply(data.matrix,2,sum))/(prod(dim(data.matrix)))

t2=sum((apply(data.matrix,2,mean)-mu.hat)^2)/(dim(data.matrix)[2])

#Computing S^2

mean.vec=apply(data.matrix,2,mean)

SS=rep(0,dim(data.matrix)[2])

for (i in 1:dim(data.matrix)[2]){

SS[i]=sum((data.matrix[,i]-mean.vec[i])^2)

}

s2=sum(SS)/prod(dim(data.matrix))

# Computing s^2* and t^2*

s2.star=(dim(data.matrix)[1]/(dim(data.matrix)[1]-1))*s2

t2.star=(dim(data.matrix)[2]/(dim(data.matrix)[2]-1))*t2

# Computing \widehat{\sigma}^2 and \widehat{d}

Z=data.matrix-mu.hat

J=matrix(1,dim(Z)[1],dim(Z)[1])

ZZ=rep(0,dim(Z)[2])

ZJZ=rep(0,dim(Z)[2])

for (i in 1:dim(Z)[2]){

ZZ[i]=t(Z[,i])%*% Z[,i]

ZJZ[i]=(t(Z[,i])%*%J)%*%Z[,i]
```

```
}

d.hat=(1/(N*n*(n-1))) * (sum(ZJZ)-sum(ZZ))

sigma2.hat=(1/(N*n*(n-1)))* ((n*sum(ZZ))-sum(ZJZ))


# computing the variance of \widehat{\mu} \widehat{\sigma}^2

  and \widehat{d}


var.mu=(sigma2.hat+ (n*d.hat))/(N*n)

var1=(2*(sigma2.hat^2))/(N*(n-1))

var12=(-1)*(2*(sigma2.hat^2))/(N*n*(n-1))

var2=(2/(n*N))*(((sigma2.hat^2)/(n-1)) + (2*sigma2.hat*d.hat)

              + (n*(d.hat^2)))

var.varcomp=matrix(c(var1,var12,var12,var2),2,2)


# computing the variance of \widehat{\mu} \widehat{\sigma}^2

  and \widehat{d}

#based on the two stage approach


var.mu.2stage=(s2+ (n*t2))/(N*n)

var.s2=(2*(n-1)*(s2^2))/(N*(n^2))
```

```
var.t2=(2*((N-1)^2))/((N^2)*n) * (((s2^2)/n) + (2*s2+t2 )
        + (n*(t2^2) ))


# computing the variance of \widehat{\mu} \widehat{\sigma}^2
  and \widehat{d}
#based on the unbiased two stage approach


var.mu.2stage.unbiased=(s2.star+ (n*t2.star))/(N*n)
var.s2.star=(2*(s2.star^2))/ (N*(n-1))
var.t2.star= (2/(N*n)) * (((s2.star^2)/n)+(2*s2.star*t2.star)
                      +(n*(t2.star^2)))


# Saving the results


mu.split.est[k]=mu.hat
Var.varcomp[[k]]=var.varcomp
mu.split.var[k,]=c(var.mu,var.mu.2stage,var.mu.2stage.unbiased)
d.split.est[k,]=c(d.hat,t2,t2.star)
sigma2.split.est[k,]=c(sigma2.hat,s2,s2.star)
d.split.var[k,]=c(var.varcomp[2,2],var.t2,var.t2.star)
```

```
sigma2.split.var[k,]=c(var.varcomp[1,1],var.s2,var.s2.star)

}

colnames(mu.split.var)=c("Weighted","TwoStage",

         "TwoStageUnbiased")

colnames(d.split.var)=c("Weighted","TwoStage",

         "TwoStageUnbiased")

colnames(sigma2.split.var)=c("Weighted","TwoStage",

         "TwoStageUnbiased")

colnames(d.split.est)=c("Weighted","TwoStage",

         "TwoStageUnbiased")

colnames(sigma2.split.est)=c("Weighted","TwoStage","TwoStageUnbiased")


return(list(mu.split.est=mu.split.est,mu.split.var=mu.split.var,

sigma2.split.est=sigma2.split.est, sigma2.split.var=sigma2.split.var,

d.split.est=d.split.est,d.split.var=d.split.var,

             Var.varcomp=Var.varcomp))}
```

References

Aerts, M., Geys, H., Molenberghs, G., and Ryan, L. (2002). *Topics in Modelling of Clustered Data*. London: Chapman & Hall.

Aerts, M., Faes, C., Hens, N., Loquiha, O., and Molenberghs, G. (2011). Incomplete clustered data and non-ignorable cluster size. In: Conesa, D., Forte, A., López-Quílez, A. and Muñoz, F. (Eds.), Proceedings of the 26th International Workshop on Statistical Modelling, València, Spain, 35–40.

Arnold, B.C. and Strauss, D. (1991) Pseudolikelihood estimation: some examples. *Sankhya B* **53**, 233–243.

Basu, D. (1955). On statistics independent of a complete sufficient statistic. *Sankhya*, **15**, 377–380.

Benhin, E., Rao, J.N.K., and Scott, A.J. (2005). Mean estimating equation approach to analysing cluster-correlated data with nonignorable cluster sizes. *Biometrika*, **92**, 435–450.

Burzykowski, T., Molenberghs, G., and Buyse, M. (2005). *The Evaluation of Surrogate Endpoints*. New York: Springer.

Casella, G. and Berger, R.L. (2001). *Statistical Inference*. Pacific Grove: Duxbury Press.

Chiang, C-T., and Lee, K-Y. (2008). Efficient estimation methods for informative cluster size data. *Statistica Sinica*, **18**, 121–133.

Cong, X-J., Yin, G., and Shen, Y. (2007). Marginal analysis of correlated failure time data with

informative cluster sizes. *Biometrics*, **63**, 663–672.

Fieuws, S. and Verbeke, G. (2006). Pairwise fitting of mixed models for the joint modelling of multivariate longitudinal profiles. *Biometrics*, **62**, 424–431.

Fieuws, S., Verbeke, G., Boen, F., and Delecluse, C. (2006). High-dimensional multivariate mixed models for binary questionnaire data. *Applied Statistics*, **55**, 1–12.

Follmann, D., Proschan, M., Leifer, E. (2003). Multiple outputation: Inference for complex Clustered data by averaging analysis form independent data. *Biometrics*, **59**, 420–429.

Hermans, L., Molenberghs, M., Aerts, M., Kenward, M.G and Verbeke, G. (2017). A tutorial on the practical use and implication of complete sufficient statistics. *Submitted*.

Hoffman, E.B., Sen, P.K., and Weinberg, C.R. (2001). Within-cluster resampling. *Biometrika*, **88**, 1121–1134.

Laird, N.M. and Ware, J.H. (1982) Random effects models for longitudinal data. *Biometrics*, **38**, 963–974.

Liu, A. and Hall, W.J. (1999). Unbiased estimation following a group sequential test. *Biometrika*, **86**, 71–78.

Liu, A., Hall, W.J., Yu, K.F., and Wu, C. (2006). Estimation following a group sequential test for distributions in the one-parameter exponential family. *Statistica Sinica*, **16**, 165–81.

Milanzi, E., Molenberghs, G., Alonso, A., Kenward, M.G., Verbeke, G., Tsiatis, A.A., and Davidian, M. (2016). Properties of estimators in exponential family settings with observation-

based stopping rules. *Journal of Biometrics & Biostatistics*, **7**, 272.

Molenberghs, G., Kenward, M.G., Aerts, M., Verbeke, G., Tsiatis, A.A., Davidian, M., Rizopoulos, D. (2014). On random sample size, ignorability, ancillarity, completeness, separability, and degeneracy: sequential trials, random sample sizes, and missing data. *Statistical Methods in Medical Research*, **23**, 11–41.

Molenberghs, G., Verbeke, G. and Iddi S. (2011). Pseudo-likelihood methodology for partitioned large and complex samples. *Statistics and probability letters*, **81**, 892–901.

Molenberghs, G. and Verbeke, G. (2005). *Models for Discrete Longitudinal Data*. New York: Springer.

Neiswanger, W., Wang, C., and Xing, E. (2013). Asymptotically exact, embarrassingly parallel MCMC. arXiv preprint arXiv:1311.4780

Price, C.J., Kimmel, C.A., George, J.D., and Marr, M.C. (1987). The developmental toxicity of diethylene glycol dimethyl ether in mice. *Fundamental and Applied Toxicology* **8**, 115–126.

Price, C.J., Kimmel, C.A., Tyl, R.W., and Marr, M.C. (1985). The developmental toxicity of ethylene glycol in rats and mice. *Toxicology and Applied Pharmacology* **81**, 113–127.

Scott, S.L., Blocker, A.W., Bonassi, F.V., Chipman, H.A., George, E.I., and McCulloch, C.E. (2013). Bayes and big data: The consensus Monte Carlo algorithm. In: Proceedings of he EFaBBayes 250 Conference, **16**.

Sikorska, K., Lesaffre, E., Groenen, P.F.J., and Eilers, P.H.C. (2013). GWAS on your notebook: fast semi-parallel linear and logistic regression for genome-wide association studies. *BMC*

bioinformatics, **14**, 166.

Tyl, R.W., Price, C.J., Marr, M.C., and Kimmel, C.A. (1988). Developmental toxicity evaluation of dietary di(2-ethylhexyl)phthalate in Fischer 344 rats and CD-1 mice. *Fundamental and Applied Toxicology* **10**, 395–412.

Varin, C., Reid, N., and Firth, D. (2011). An overview of composite likelihood methods. *Statistica Sinica*, **21**, 5–42.

Van der Elst, W., Hermans, L., Verbeke, G., Kenward, M., Nassiri, V. and Molenberghs, G. (2015). Unbalanced cluster sizes and rates of convergence in mixed-effects models for clustered data. *Journal of Statistical Computation and Simulation*, **86(11)**, 1–17.

Verbeke, G., and Molenberghs, G. (2000). *Linear Mixed Models for Longitudinal Data*. New York: Springer.

Wang, M., Kong, M., and Datta, S. (2011). Inference for marginal linear models for correlated longitudinal data with potentially informative cluster sizes. *Statistical Methods in Medical Research*, **20**, 347–367.

Williamson, J.M., Datta, S., and Satten, G.A. (2003). Marginal analyses of clustered data when cluster size is informative. *Biometrics*, **59**, 36–42.