Reproducibility of grouped data assessed by intraclass correlation coefficient - eAppendix

1 Intraclass correlation coefficient as a reproducibility index

In a reproducibility study, $M$ respondents complete the same questionnaire on two different occasions. In terms of a random effects model, we write

$$Y_{ij} = \mu + \beta_i + \epsilon_{ij}, \quad i = 1, 2, \ldots M, \quad j = 1, 2. \tag{1}$$

Here, $\mu$ represents the overall mean, $\beta_i$ is the deviation of respondent $i$ from the overall mean, and $\epsilon_{ij}$ is the deviation of $i$-th respondent’s reported value from his true value, $\mu + \beta_i$, on occasion $j$. $\beta_i$ and $\epsilon_{ij}$ are independent normal with mean 0 and variance $\sigma_\beta^2$ and $\sigma^2$, respectively.

The intraclass correlation coefficient (ICC) $\rho$ is calculated as share of between-class variability $\sigma_\beta^2$ in total variability, $\rho = \sigma_\beta^2/(\sigma_\beta^2 + \sigma^2)$. Defined as a ratio of variances, it ranges from 0 to 1, with higher values indicating better reproducibility.

In the case of continuous data, parameters $\mu$, $\sigma_\beta^2$ and $\sigma^2$ are obtained by maximizing the log-likelihood function. For grouped data, we derive likelihood in the following section. Optimization details are given in sections 3 and 5. We provide implementation in the R package iRepro (available from http://www.imi.hr/~jkovacic/irepro.html).

2 Derivation of the log-likelihood for grouped data

Following Pinheiro and Bates (2000), the likelihood for model (1) given continuous data $y$ is expressed as

$$L(\mu, \sigma_\beta^2, \sigma^2|y) = \prod_{i=1}^{M} \int_{-\infty}^{\infty} P(y_{i1}|\beta_i, \mu, \sigma^2) P(\beta_i|\sigma_\beta^2, \mu, \sigma^2) d\beta_i,$$

$$= \prod_{i=1}^{M} \int_{-\infty}^{\infty} \frac{1}{2\pi\sigma^2} \exp \left( -\frac{\|y_{i1} - \frac{1}{2}\mu - \frac{1}{2}\beta_i\|^2}{2\sigma^2} \right) \frac{1}{\sqrt{2\pi\sigma_\beta^2}} \exp \left( -\frac{\beta_i^2}{2\sigma_\beta^2} \right) d\beta_i. \tag{2}$$

In the case of grouped data, we observe only $a_i \leq y_{ij} < b_i$ (corresponding to category „not less than $a_i$, but less than $b_i$”) and $c_i \leq y_{ij} < d_i$. In the same manner as before, we derive the likelihood for such data. Let $X$ and $Z$ denote design matrices for fixed and random effects, respectively; in the case of one-way random effects model (1) both are equal to $2 \times 1$ matrix of ones, $[1 \, 1]^T$. If we denote $Y = [y_{11} \, y_{12}]^T$, we can write the likelihood for the $i$-th respondent as

$$L_i = \int_{-\infty}^{\infty} \int_{a_i}^{b_i} \frac{1}{2\pi\sigma^2} \exp \left( -\frac{\|Y - X_{i\mu} - Z_i\beta_i\|^2}{2\sigma^2} \right) \frac{1}{\sqrt{2\pi\sigma_\beta^2}} \exp \left( -\frac{\beta_i^2}{2\sigma_\beta^2} \right) dy_{1j}dy_{2j}d\beta_i.$$

The same derivation applies in cases when $a_i < y_{ij} < b_i$, $a_i < y_{ij} \leq b_i$ or $a_i \leq y_{ij} \leq b_i$. If we denote $\hat{\beta}_i = (Z^TZ)^{-1}Z^T(Y - X\mu)$, we can substitute $\|Y - X_{i\mu} - Z_i\beta_i\|^2$ with $\|Y - X\mu - Z\hat{\beta}_i\|^2 + \|Z(\beta_i - \hat{\beta}_i)\|^2$. Then, using
We note that the inner integral can be reduced to integral of normal density function, which is then easily solved:

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi \sigma^2}} \exp \left( -\frac{\beta_i^2}{2\sigma^2} - \frac{||Z(\beta_i - \bar{\beta}_i)||^2}{2\sigma^2} \right) d\beta_i dy_1. \quad (3)$$

We note that the inner integral can be reduced to integral of normal density function, which is then easily solved:

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi \sigma^2}} \exp \left( -\frac{\beta_i^2}{2\sigma^2} - \frac{||Z(\beta_i - \bar{\beta}_i)||^2}{2\sigma^2} \right) d\beta_i dy_1. \quad (4)$$

Here $\Delta^2$ stands for $\sigma^2/\sigma^2$. Substituting the inner integral in (3) with (4), the expression $L_i$ reduces to

$$\frac{1}{2\pi \sigma^2} \frac{\Delta^2}{\Delta^2 + 2} \int_{a_i, c_i}^{b_i, d_i} \exp \left( -\frac{u}{\sigma^2} \right) \exp \left( -\frac{\Delta^2}{\Delta^2 + 2} \frac{(v - \mu)^2}{\sigma^2} \right) dv du,$$

where $D$ is the area bounded by the lines $v = -u + a_i$, $v = u + c_i$, $v = -u + b_i$, and $v = u + d_i$. Integrating first with respect to $v$ and afterward with respect to $u$, taking the natural logarithm, and summing contributions for each respondent results in the log-likelihood

$$C + \sum_{i=1}^{M} \log L_i',$$

where $L_i'$ equals to

$$\int_{a_i, c_i}^{b_i, d_i} \exp \left( -\frac{u}{\sigma^2} \right) \Phi \left( \frac{u + d_i - \mu}{\sqrt{0.5\sigma^2 + \sigma^2}} \right) du + \int_{a_i, c_i}^{b_i, d_i} \exp \left( -\frac{u^2}{\sigma^2} \right) \Phi \left( \frac{-u + b_i - \mu}{\sqrt{0.5\sigma^2 + \sigma^2}} \right) du$$

$$- \int_{a_i, c_i}^{b_i, d_i} \exp \left( -\frac{u}{\sigma^2} \right) \Phi \left( \frac{-u + a_i - \mu}{\sqrt{0.5\sigma^2 + \sigma^2}} \right) du - \int_{a_i, c_i}^{b_i, d_i} \exp \left( -\frac{u^2}{\sigma^2} \right) \Phi \left( \frac{u + c_i - \mu}{\sqrt{0.5\sigma^2 + \sigma^2}} \right) du. \quad (5)$$

Here $C$ equals to $-(M/2) \log \pi - M \log \sigma$, while $\Phi$ denotes the cumulative distribution function of a standard normal distribution. We can rewrite the expression (5) in a slightly different manner, as this will be beneficial in an optimization procedure:

$$\int_{a_i, c_i}^{b_i, d_i} \exp \left( -\frac{u^2}{\sigma^2} \right) \left( \Phi \left( \frac{u + d_i - \mu}{\sqrt{0.5\sigma^2 + \sigma^2}} \right) - \Phi \left( \frac{-u + b_i - \mu}{\sqrt{0.5\sigma^2 + \sigma^2}} \right) \right) du$$

$$+ \int_{a_i, c_i}^{b_i, d_i} \exp \left( -\frac{u^2}{\sigma^2} \right) \left( \Phi \left( \frac{-u + a_i - \mu}{\sqrt{0.5\sigma^2 + \sigma^2}} \right) - \Phi \left( \frac{u + c_i - \mu}{\sqrt{0.5\sigma^2 + \sigma^2}} \right) \right) du + l_i^{(i)} \quad (6)$$
where \( l_1(i) = \frac{(b_i - d_i)}{2} \) if \( b_i - d_i \leq a_i - c_i \) and \( (a_i - c_i)/2 \) otherwise; \( l_2(i) = \frac{(b_i - d_i)}{2} \) if \( a_i - c_i \leq b_i - d_i \) and \( (a_i - c_i)/2 \) otherwise; and

\[
l_3(i) = \begin{cases} \frac{a_i - c_i}{2} & \text{if } a_i - c_i > b_i - d_i, \\
\int_{\frac{b_i - d_i}{2}}^{\frac{b_i - d_i}{2}} \exp\left(-\frac{u^2}{\sigma^2}\right) \left(\Phi\left(-\frac{u + b_i - \mu}{\sqrt{0.5\sigma^2 + \beta^2}}\right) - \Phi\left(-\frac{u + d_i - \mu}{\sqrt{0.5\sigma^2 + \beta^2}}\right)\right) du, & \text{if } a_i - c_i > b_i - d_i, \\
\int_{\frac{b_i - d_i}{2}}^{\frac{a_i - c_i}{2}} \exp\left(-\frac{u^2}{\sigma^2}\right) \left(\Phi\left(-\frac{u + b_i - \mu}{\sqrt{0.5\sigma^2 + \beta^2}}\right) - \Phi\left(-\frac{u + c_i - \mu}{\sqrt{0.5\sigma^2 + \beta^2}}\right)\right) du, & \text{if } a_i - c_i < b_i - d_i, \\
0 & \text{otherwise.}
\end{cases}
\]

3 Methodology of simulations

For a fixed value \( \rho \) of the ICC, we sampled \( \sigma_\beta \) from uniform distribution in the range from 0 to 50, and then calculated \( \sigma_\beta \) as \( \sigma_\beta \sqrt{(1 - \rho)/\rho} \). Parameter \( \mu \) was fixed at zero in all experiments.

For each experiment, we calculated estimators’ means and standard deviations (SD) over 1000 simulations. Besides midpoint ICC calculated on the midpoints of categories (ICC\textsubscript{MID}) and maximum likelihood ICC (ICC\textsubscript{MLE}) calculated by maximizing (5), the results include continuous ICC (ICC\textsubscript{C}), calculated on original (uncensored) data as a reference. We used restricted maximum likelihood estimation as a gold standard for uncensored data.

All simulations were performed in the free statistical software R, version 2.13.2 (R Foundation for Statistical Computing, Vienna, Austria). Random effects models were estimated using the \texttt{nlme} package.\textsuperscript{2} Maximum likelihood estimates for grouped data were obtained by linearly constrained optimization (\texttt{constrOptim}) using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method. Results were obtained by maximizing (5) with a gradient supplied. In the case of unequal categories widths, we additionally performed a part of the simulations by maximizing (6), as it proved to be more numerically stable in this case (see section 5 for further details). The code used in simulations is given in section 7.

4 Results of simulations

eTable 1 and eFigure 1 show results of simulations with 5 categories. Simulations for different number of categories are summarized in eTables 2-4.

table 1. Results of Simulations with 5 Categories

<table>
<thead>
<tr>
<th>( \rho )</th>
<th>ICC\textsuperscript{a} Mean(SD)</th>
<th>ICC\textsubscript{MID}\textsuperscript{b} Mean(SD)</th>
<th>ICC\textsubscript{MLE}\textsuperscript{c} Mean(SD)</th>
<th>ICC\textsubscript{C} Mean(SD)</th>
<th>ICC\textsubscript{MID} Mean(SD)</th>
<th>ICC\textsubscript{MLE} Mean(SD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.101(0.032)</td>
<td>0.088(0.031)</td>
<td>0.101(0.036)</td>
<td>0.100(0.031)</td>
<td>0.068(0.035)</td>
<td>0.101(0.049)</td>
</tr>
<tr>
<td>0.2</td>
<td>0.199(0.031)</td>
<td>0.171(0.031)</td>
<td>0.197(0.036)</td>
<td>0.200(0.031)</td>
<td>0.136(0.043)</td>
<td>0.199(0.050)</td>
</tr>
<tr>
<td>0.3</td>
<td>0.300(0.028)</td>
<td>0.259(0.029)</td>
<td>0.300(0.033)</td>
<td>0.301(0.028)</td>
<td>0.208(0.051)</td>
<td>0.299(0.048)</td>
</tr>
<tr>
<td>0.4</td>
<td>0.399(0.027)</td>
<td>0.346(0.028)</td>
<td>0.400(0.032)</td>
<td>0.401(0.026)</td>
<td>0.283(0.059)</td>
<td>0.400(0.049)</td>
</tr>
<tr>
<td>0.5</td>
<td>0.500(0.023)</td>
<td>0.433(0.026)</td>
<td>0.501(0.028)</td>
<td>0.500(0.024)</td>
<td>0.354(0.067)</td>
<td>0.497(0.042)</td>
</tr>
<tr>
<td>0.6</td>
<td>0.601(0.021)</td>
<td>0.519(0.025)</td>
<td>0.601(0.026)</td>
<td>0.600(0.020)</td>
<td>0.431(0.075)</td>
<td>0.597(0.038)</td>
</tr>
<tr>
<td>0.7</td>
<td>0.700(0.016)</td>
<td>0.605(0.022)</td>
<td>0.700(0.022)</td>
<td>0.699(0.016)</td>
<td>0.522(0.072)</td>
<td>0.698(0.032)</td>
</tr>
<tr>
<td>0.8</td>
<td>0.800(0.012)</td>
<td>0.694(0.019)</td>
<td>0.800(0.016)</td>
<td>0.800(0.012)</td>
<td>0.625(0.068)</td>
<td>0.800(0.024)</td>
</tr>
<tr>
<td>0.9</td>
<td>0.900(0.006)</td>
<td>0.791(0.016)</td>
<td>0.900(0.011)</td>
<td>0.900(0.006)</td>
<td>0.742(0.056)</td>
<td>0.899(0.015)</td>
</tr>
</tbody>
</table>

\textsuperscript{a} ICC of continuous data (included as a reference).

\textsuperscript{b} ICC of grouped data, calculated on midpoints of categories.

\textsuperscript{c} ICC of grouped data, estimated by maximum likelihood.
eFigure1: Estimator performance in the case of 5 categories. For each of the 99 ICC values (0.01 - 0.99), plot shows means of ICC\(_{\text{MID}}\) and ICC\(_{\text{MLE}}\) over 1000 simulations with equal category widths and 1000 simulations with unequal category widths. True ICC value is shown as the identity line. ICC\(_{\text{MID}}\): ICC calculated on categories’ midpoints; ICC\(_{\text{MLE}}\): maximum likelihood estimator.

eTable 2. Effect of Number of Categories on Estimator Performance for ICC = 0.2

<table>
<thead>
<tr>
<th>Categories</th>
<th>ICC(_C^a) Mean(SD)</th>
<th>ICC(_{\text{MID}}^b) Mean(SD)</th>
<th>ICC(_{\text{MLE}}^c) Mean(SD)</th>
<th>ICC(_C^a) Mean(SD)</th>
<th>ICC(_{\text{MID}}^b) Mean(SD)</th>
<th>ICC(_{\text{MLE}}^c) Mean(SD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.199(0.029)</td>
<td>0.136(0.031)</td>
<td>0.198(0.044)</td>
<td>0.200(0.030)</td>
<td>0.107(0.049)</td>
<td>0.195(0.077)</td>
</tr>
<tr>
<td>5</td>
<td>0.200(0.030)</td>
<td>0.173(0.030)</td>
<td>0.200(0.035)</td>
<td>0.200(0.030)</td>
<td>0.136(0.044)</td>
<td>0.199(0.052)</td>
</tr>
<tr>
<td>10</td>
<td>0.201(0.030)</td>
<td>0.193(0.030)</td>
<td>0.200(0.032)</td>
<td>0.201(0.029)</td>
<td>0.171(0.035)</td>
<td>0.201(0.036)</td>
</tr>
<tr>
<td>25</td>
<td>0.198(0.031)</td>
<td>0.197(0.031)</td>
<td>0.198(0.031)</td>
<td>0.198(0.030)</td>
<td>0.191(0.030)</td>
<td>0.198(0.031)</td>
</tr>
<tr>
<td>50</td>
<td>0.199(0.031)</td>
<td>0.198(0.031)</td>
<td>0.198(0.031)</td>
<td>0.201(0.030)</td>
<td>0.199(0.030)</td>
<td>0.200(0.030)</td>
</tr>
</tbody>
</table>

\(a\) ICC of continuous data (included as a reference).
\(b\) ICC of grouped data, calculated on midpoints of categories.
\(c\) ICC of grouped data, estimated by maximum likelihood.

eTable 3. Effect of Number of Categories on Estimator Performance for ICC = 0.5

<table>
<thead>
<tr>
<th>Categories</th>
<th>ICC(_C^a) Mean(SD)</th>
<th>ICC(_{\text{MID}}^b) Mean(SD)</th>
<th>ICC(_{\text{MLE}}^c) Mean(SD)</th>
<th>ICC(_C^a) Mean(SD)</th>
<th>ICC(_{\text{MID}}^b) Mean(SD)</th>
<th>ICC(_{\text{MLE}}^c) Mean(SD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.499(0.024)</td>
<td>0.344(0.031)</td>
<td>0.499(0.039)</td>
<td>0.500(0.023)</td>
<td>0.296(0.079)</td>
<td>0.495(0.078)</td>
</tr>
<tr>
<td>5</td>
<td>0.500(0.023)</td>
<td>0.432(0.026)</td>
<td>0.500(0.028)</td>
<td>0.499(0.024)</td>
<td>0.354(0.070)</td>
<td>0.497(0.048)</td>
</tr>
<tr>
<td>10</td>
<td>0.500(0.024)</td>
<td>0.481(0.025)</td>
<td>0.500(0.026)</td>
<td>0.499(0.023)</td>
<td>0.429(0.050)</td>
<td>0.499(0.029)</td>
</tr>
<tr>
<td>25</td>
<td>0.501(0.024)</td>
<td>0.498(0.024)</td>
<td>0.501(0.024)</td>
<td>0.501(0.023)</td>
<td>0.484(0.026)</td>
<td>0.500(0.024)</td>
</tr>
<tr>
<td>50</td>
<td>0.500(0.023)</td>
<td>0.499(0.023)</td>
<td>0.500(0.023)</td>
<td>0.499(0.024)</td>
<td>0.495(0.024)</td>
<td>0.499(0.024)</td>
</tr>
</tbody>
</table>

\(a\) ICC of continuous data (included as a reference).
\(b\) ICC of grouped data, calculated on midpoints of categories.
\(c\) ICC of grouped data, estimated by maximum likelihood.
TABLE 4. Effect of Number of Categories on Estimator Performance for ICC = 0.8

<table>
<thead>
<tr>
<th>Categories</th>
<th>Categories of equal widths</th>
<th>Categories of unequal widths</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ICC(^a) Mean(SD)</td>
<td>ICC(^a) Mean(SD)</td>
</tr>
<tr>
<td>3</td>
<td>0.800(0.011)</td>
<td>0.591(0.028)</td>
</tr>
<tr>
<td>5</td>
<td>0.800(0.011)</td>
<td>0.694(0.019)</td>
</tr>
<tr>
<td>10</td>
<td>0.799(0.011)</td>
<td>0.769(0.013)</td>
</tr>
<tr>
<td>25</td>
<td>0.800(0.012)</td>
<td>0.795(0.012)</td>
</tr>
<tr>
<td>50</td>
<td>0.800(0.011)</td>
<td>0.790(0.011)</td>
</tr>
</tbody>
</table>

\(^a\) ICC of continuous data (included as a reference).
\(^b\) ICC of grouped data, calculated on midpoints of categories.
\(^c\) ICC of grouped data, estimated by maximum likelihood.

5 Numerical optimization: details and issues

When widths of categories were equal, midpoint method (ICC\(_{\text{MID}}\)) and maximum likelihood estimator (ICC\(_{\text{MLE}}\)) performed similarly in terms of failure rate. The proportion of cases when each of the estimators failed to estimate ICC was less than 2% under all conditions. In the case of unequal widths, ICC\(_{\text{MID}}\) kept the failure rate under 2%. ICC\(_{\text{MLE}}\), however, dramatically worsened. For ICC values from 0.01 to 0.9, the number of failure cases per 1000 successful simulations increased at a slow rate in the range from 50 to 218, proportionally to the increase of ICC values. At ICC values greater than 0.9, the number of failures suddenly worsened, reaching a maximum value of 3206 (76%) for the highest value of ICC, 0.99.

After examining the failure cases, we concluded that the problem occurred when numerically integrating highly-peaked Gaussian functions over small area regions far from the mean of distribution. This procedure results in very small and imprecise probability estimates (less than machine precision, often less than \(10^{-100}\)). Consequently, the subtraction of such small-valued integrals led to an erroneous result (namely, to a probability estimate less than zero). This effect is known in numerical mathematics as catastrophic cancellation.

For this reason, we ran additional simulations by optimizing (6), a mathematically equivalent expression to (5). By using this method, catastrophic cancellation is avoided since integrals are summed and no subtraction of small numbers occurs. The results are shown in eFigure 2. In terms of bias, it performed slightly worse than the estimator obtained by optimizing (5) - for ICC values greater than 0.1, the difference between its mean and true ICC value ranged from 0.000 to 0.013 (median value of 0.005). On the other hand, it had a much lower failure rate. Number of failure cases per 1000 successful simulations was no higher than 45 (4%).
Unequal category widths:

\[ - \text{ICC} \quad - \text{MID} \quad - \text{MLE} \]

6 Figure 2: Estimator performance in the case of 5 categories when maximum likelihood estimator is obtained by optimizing (6). For each of the 99 ICC values (0.01 - 0.99), plot shows means of ICC_{MID} and ICC_{MLE} over 1000 simulations with equal category widths and 1000 simulations with unequal category widths. True ICC value is shown as the identity line. ICC_{MID}: ICC calculated on categories’ midpoints; ICC_{MLE}: maximum likelihood estimator.

6 Example: food frequency questionnaire

Apart from simulations, we compared maximum likelihood ICC and midpoint ICC on real data. For this purpose we use the example of food frequency questionnaire (FFQ), the most common dietary assessment tool. The data originate from the study investigating genetic, environmental, and lifestyle factors, including diet, related to atopic respiratory and skin diseases in adolescents.\(^3\) The reproducibility of the FFQ, designed to assess the intake of nutrients suspected to be related to atopy, was evaluated in 106 subjects, although for some food items this number was lower. The subjects completed the same questionnaire twice, on average three months apart.

There were six predefined categories for frequency of food intake: never/less than once a month, one to three times a month, two to four times a week, five to six times a week, and every day. For the purpose of ICC calculation, we expressed them as intervals of daily intake (e.g., the first category, never/less than once a month, was expressed as interval \([0, \frac{1}{30}])\). If there were gaps between two neighbouring intervals, we extended both intervals in such a way that each covered half of gap. In cases when data were skewed, we transformed them by adding a constant and taking the natural logarithm prior to calculation of the ICC.

The midpoint ICC for log-transformed data was calculated in two ways: on transformed midpoints (ICC_{MID1}), and on midpoints of transformed categories (ICC_{MID2}). The results were very similar (eTable 5). We additionally showed the values of Pearson’s and Spearman’s correlation coefficients, as they are usually calculated in parallel to ICC, and kappa with linear and quadratic weights. The mean value of ICC_{MLE} was 0.67 (range 0.41 - 0.82). In comparison, the mean of both ICC_{MID1} and ICC_{MID2} was 0.58, with their values ranging from 0.37 to 0.76 and 0.37 to 0.75, respectively. The mean value of both Pearson’s and Spearman’s correlation was 0.59. Mean kappa coefficient was 0.48 for linear weights and 0.58 for quadratic weights. As in the simulations, ICC_{MLE} estimates were constantly higher than that of ICC_{MID1} and ICC_{MID2}. The mean of their absolute differences was 0.09 (range 0.04 - 0.18) for both ICC_{MID1} and ICC_{MID2}. 
<table>
<thead>
<tr>
<th>Food item</th>
<th>Pearson&lt;sup&gt;a&lt;/sup&gt;</th>
<th>Spearman&lt;sup&gt;b&lt;/sup&gt;</th>
<th>ICC&lt;sub&gt;MID1&lt;/sub&gt;&lt;sup&gt;c&lt;/sup&gt;</th>
<th>ICC&lt;sub&gt;MID2&lt;/sub&gt;&lt;sup&gt;d&lt;/sup&gt;</th>
<th>ICC&lt;sub&gt;MLE&lt;/sub&gt;&lt;sup&gt;e&lt;/sup&gt;</th>
<th>κ&lt;sub&gt;L&lt;/sub&gt;&lt;sup&gt;f&lt;/sup&gt;</th>
<th>κ&lt;sub&gt;Q&lt;/sub&gt;&lt;sup&gt;g&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cured meat</td>
<td>0.54</td>
<td>0.56</td>
<td>0.54</td>
<td>0.54</td>
<td>0.59</td>
<td>0.40</td>
<td>0.54</td>
</tr>
<tr>
<td>Milk, yoghurt</td>
<td>0.65</td>
<td>0.67</td>
<td>0.65</td>
<td>0.63</td>
<td>0.78</td>
<td>0.58</td>
<td>0.65</td>
</tr>
<tr>
<td>Cheese</td>
<td>0.58</td>
<td>0.58</td>
<td>0.58</td>
<td>0.58</td>
<td>0.66</td>
<td>0.43</td>
<td>0.58</td>
</tr>
<tr>
<td>Cottage cheese</td>
<td>0.77</td>
<td>0.75</td>
<td>0.76</td>
<td>0.75</td>
<td>0.82</td>
<td>0.62</td>
<td>0.76</td>
</tr>
<tr>
<td>Butter</td>
<td>0.60</td>
<td>0.59</td>
<td>0.60</td>
<td>0.58</td>
<td>0.71</td>
<td>0.52</td>
<td>0.59</td>
</tr>
<tr>
<td>Margarine</td>
<td>0.54</td>
<td>0.54</td>
<td>0.55</td>
<td>0.56</td>
<td>0.66</td>
<td>0.46</td>
<td>0.54</td>
</tr>
<tr>
<td>Sunflower oil</td>
<td>0.37</td>
<td>0.38</td>
<td>0.37</td>
<td>0.37</td>
<td>0.41</td>
<td>0.25</td>
<td>0.36</td>
</tr>
<tr>
<td>Eggs</td>
<td>0.54</td>
<td>0.61</td>
<td>0.54</td>
<td>0.54</td>
<td>0.72</td>
<td>0.47</td>
<td>0.54</td>
</tr>
<tr>
<td>Legumes</td>
<td>0.39</td>
<td>0.43</td>
<td>0.38</td>
<td>0.39</td>
<td>0.44</td>
<td>0.33</td>
<td>0.38</td>
</tr>
<tr>
<td>Soya products</td>
<td>0.71</td>
<td>0.77</td>
<td>0.68</td>
<td>0.74</td>
<td>0.81</td>
<td>0.61</td>
<td>0.68</td>
</tr>
<tr>
<td>Offal</td>
<td>0.61</td>
<td>0.63</td>
<td>0.61</td>
<td>0.63</td>
<td>0.77</td>
<td>0.58</td>
<td>0.61</td>
</tr>
<tr>
<td>Blue fish</td>
<td>0.56</td>
<td>0.57</td>
<td>0.56</td>
<td>0.55</td>
<td>0.61</td>
<td>0.46</td>
<td>0.56</td>
</tr>
<tr>
<td>White fish</td>
<td>0.60</td>
<td>0.55</td>
<td>0.60</td>
<td>0.59</td>
<td>0.64</td>
<td>0.52</td>
<td>0.60</td>
</tr>
<tr>
<td>Seashells, shrimps, crustaceans etc.</td>
<td>0.57</td>
<td>0.57</td>
<td>0.55</td>
<td>0.56</td>
<td>0.63</td>
<td>0.48</td>
<td>0.55</td>
</tr>
<tr>
<td>Potatoes</td>
<td>0.61</td>
<td>0.60</td>
<td>0.60</td>
<td>0.60</td>
<td>0.74</td>
<td>0.48</td>
<td>0.60</td>
</tr>
<tr>
<td>Pasta</td>
<td>0.51</td>
<td>0.48</td>
<td>0.52</td>
<td>0.53</td>
<td>0.67</td>
<td>0.38</td>
<td>0.51</td>
</tr>
<tr>
<td>Cooked leafy vegetable</td>
<td>0.59</td>
<td>0.58</td>
<td>0.58</td>
<td>0.58</td>
<td>0.64</td>
<td>0.43</td>
<td>0.58</td>
</tr>
<tr>
<td>Raw leafy vegetable</td>
<td>0.55</td>
<td>0.55</td>
<td>0.52</td>
<td>0.52</td>
<td>0.57</td>
<td>0.40</td>
<td>0.53</td>
</tr>
<tr>
<td>Berries</td>
<td>0.59</td>
<td>0.65</td>
<td>0.58</td>
<td>0.58</td>
<td>0.62</td>
<td>0.49</td>
<td>0.58</td>
</tr>
<tr>
<td>Kiwi</td>
<td>0.52</td>
<td>0.56</td>
<td>0.51</td>
<td>0.58</td>
<td>0.66</td>
<td>0.46</td>
<td>0.51</td>
</tr>
<tr>
<td>Dried fruit</td>
<td>0.66</td>
<td>0.66</td>
<td>0.66</td>
<td>0.67</td>
<td>0.74</td>
<td>0.56</td>
<td>0.65</td>
</tr>
<tr>
<td>Nuts</td>
<td>0.54</td>
<td>0.50</td>
<td>0.54</td>
<td>0.43</td>
<td>0.61</td>
<td>0.39</td>
<td>0.54</td>
</tr>
<tr>
<td>Honey</td>
<td>0.56</td>
<td>0.56</td>
<td>0.56</td>
<td>0.53</td>
<td>0.61</td>
<td>0.44</td>
<td>0.56</td>
</tr>
<tr>
<td>Dark chocolate</td>
<td>0.55</td>
<td>0.57</td>
<td>0.55</td>
<td>0.56</td>
<td>0.64</td>
<td>0.47</td>
<td>0.55</td>
</tr>
<tr>
<td>Green and white tea</td>
<td>0.62</td>
<td>0.64</td>
<td>0.62</td>
<td>0.64</td>
<td>0.73</td>
<td>0.57</td>
<td>0.62</td>
</tr>
<tr>
<td>Red wine</td>
<td>0.67</td>
<td>0.65</td>
<td>0.67</td>
<td>0.65</td>
<td>0.78</td>
<td>0.56</td>
<td>0.67</td>
</tr>
<tr>
<td>White wine</td>
<td>0.68</td>
<td>0.69</td>
<td>0.68</td>
<td>0.69</td>
<td>0.82</td>
<td>0.63</td>
<td>0.68</td>
</tr>
<tr>
<td>Strong drinks</td>
<td>0.57</td>
<td>0.58</td>
<td>0.57</td>
<td>0.56</td>
<td>0.68</td>
<td>0.47</td>
<td>0.56</td>
</tr>
<tr>
<td>Carbonated beverages</td>
<td>0.74</td>
<td>0.74</td>
<td>0.72</td>
<td>0.72</td>
<td>0.79</td>
<td>0.57</td>
<td>0.72</td>
</tr>
<tr>
<td>Potato chips, saltsticks, peanuts etc.</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
<td>0.68</td>
<td>0.44</td>
<td>0.62</td>
</tr>
<tr>
<td>Mean</td>
<td>0.59</td>
<td>0.59</td>
<td>0.58</td>
<td>0.58</td>
<td>0.67</td>
<td>0.48</td>
<td>0.58</td>
</tr>
</tbody>
</table>

<sup>a</sup> Pearson’s correlation coefficient. For log-transformed data, it was calculated on log-transformed midpoints.

<sup>b</sup> Spearman’s correlation coefficient. For log-transformed data, it was calculated on log-transformed midpoints.

<sup>c</sup> ICC calculated on log-transformed midpoints of categories.

<sup>d</sup> ICC calculated on midpoints of log-transformed categories.

<sup>e</sup> ICC estimated by maximum likelihood.

<sup>f</sup> Kappa with linear weights. For categories i and j with midpoints \( m_i \) and \( m_j \), weight was calculated as \( w_{ij} = 1 - |m_i - m_j| / \max_{i',j'} |m_{i'} - m_{j'}| \).

For log-transformed data, log-transformed midpoints were used.

<sup>g</sup> Kappa with quadratic weights. For categories i and j with midpoints \( m_i \) and \( m_j \), weight was calculated as \( w_{ij} = 1 - (m_i - m_j)^2 / \max_{i',j'} (m_{i'} - m_{j'})^2 \).

For log-transformed data, log-transformed midpoints were used.
7 Code

The code used in this paper consists of functions used in maximum likelihood estimation (7.1) and simulation code (7.2). We describe how to use the provided code to estimate ICC from grouped data in 7.3. All code is written in the R software.

Prior to running the simulation code, functions from 7.1 and the package nlme need to be loaded into the R workspace. The user can vary ICC values, number of respondents, number of classes, and choose between equal and unequal widths of categories to obtain results presented in the paper.

All functions are incorporated in the R package iRepro.

7.1 Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>intervalICC</td>
<td>main function for estimating ICC from grouped data</td>
</tr>
<tr>
<td>f.int</td>
<td>auxiliary function, used in likelihood calculation</td>
</tr>
<tr>
<td>fmod.int</td>
<td>auxiliary function, used in likelihood calculation</td>
</tr>
<tr>
<td>sigma.int</td>
<td>auxiliary function, used in likelihood calculation</td>
</tr>
<tr>
<td>sigmab.int</td>
<td>auxiliary function, used in likelihood calculation</td>
</tr>
<tr>
<td>beta.i</td>
<td>auxiliary function, used in likelihood calculation</td>
</tr>
<tr>
<td>cfun1</td>
<td>cost function for optimizing log-likelihood</td>
</tr>
<tr>
<td>cfun2</td>
<td>cost function for optimizing log-likelihood given by (6); more numerically stable than cfun1</td>
</tr>
<tr>
<td>gradcfun1</td>
<td>gradient of cfun1</td>
</tr>
<tr>
<td>gradcfun2</td>
<td>gradient of cfun2</td>
</tr>
<tr>
<td>intervalICC.est1</td>
<td>auxiliary function for optimizing cfun1</td>
</tr>
<tr>
<td>intervalICC.est2</td>
<td>auxiliary function for optimizing cfun2</td>
</tr>
</tbody>
</table>

intervalICC <- function(ratings1, ratings2, classes, c.limits, optim.method=1){

# Check if all arguments are correctly specified
if(missing(ratings1) | missing(ratings2))
  stop("Unspecified ratings")
if(optim.method!=1 & optim.method!=2)
  stop("Misspecified optimization method: must be 1 or 2")
if(missing(classes))
  stop("Unspecified classes")
if(missing(c.limits))
  stop("Unspecified classes limits")
if(!is.matrix(c.limits) & !is.data.frame(c.limits))
  stop("Classes limits should be a matrix or a data frame")
if(!is.vector(classes))
  stop("Classes should be a vector")
if(!is.vector(ratings1) | !is.vector(ratings2))
  stop("Ratings1 and ratings2 should be vectors")
if(length(ratings1)!=length(ratings2))
  stop("Ratings1 and ratings2 should have equal lengths")

ratings <- as.data.frame(cbind(ratings1,ratings2))
c.limits <- as.data.frame(c.limits)

if(ncol(c.limits)!=2)
  stop("Classes limits should be a matrix or a data frame with 2 columns")
if(length(classes)!=nrow(c.limits))
  stop("Number of classes differs from number of intervals given in c.limits")
names(ratings) <- c("t1","t2")
names(c.limits) <- c("lower","upper")

if(!is.numeric(c.limits$lower) | !is.numeric(c.limits$upper))
  stop("Classes limits must be numeric")
if(any(is.na(classes)))
  stop("Missing values in classes")
if(any(is.na(c.limits)))
  stop("Missing values in classes limits")
if(any(is.na(ratings))){
  warning("Missing values in ratings detected: data rows omitted from calculation")
  ratings <- na.omit(ratings)
}

ratings$t1 <- factor(ratings$t1, levels=classes)
ratings$t2 <- factor(ratings$t2, levels=classes)

if(any(is.na(ratings))){
  stop("Unrecognized class in ratings")
if(any(c.limits$lower >= c.limits$upper))
  stop("Misspecified classes limits: lower bound equal to upper bound or greater detected")

  c.means <- rowMeans(c.limits)
  n.classes <- nrow(c.limits)
  n.resp <- nrow(ratings)

  t.means <- mat.or.vec(n.classes,n.classes)
  t.sd <- mat.or.vec(n.classes,n.classes)
  for(i in 1:n.classes){
    for(j in i:n.classes){
      t.means[i,j] <- 0.5*(c.means[i] + c.means[j])
      t.means[j,i] <- t.means[i,j]
      t.sd[i,j] <- sqrt((c.means[i] - t.means[i,j])^2 + (c.means[j] - t.means[i,j])^2)
      t.sd[j,i] <- t.sd[i,j]
    }
  }

  t.r <- table(ratings$t1,ratings$t2)
  theta0 <- c(0,0, sum(t.means*t.r)/n.resp)
  theta0[1:2] <- c(max(sqrt(sum((t.means - theta0[3])^2*t.r)/(n.resp-1))),
  .Machine$double.eps+1e-10), max(sum(t.sd*t.r)/n.resp,
  .Machine$double.eps+1e-10))

  if(optim.method==1){
    est <- intervalICC.est1(ratings,classes,c.limits,theta0)
  }else{
    est <- intervalICC.est2(ratings,classes,c.limits,theta0)
  }

  class(est) <- "ICCfit"
  est
}
f.int <- function(z, sgn, params, sd.t, lower.limit, upper.limit){
  I <- function(u){
    return(exp(-((u/params[1])^2)*pnorm((sgn*u + z - params[3])/sd.t)));
  }
  return(integrate(I, lower.limit, upper.limit, stop.on.error=1)$value);
}

fmod.int <- function(z1, sgn1, z2, sgn2, params, sd.t, lower.limit, upper.limit){
  I <- function(u){
    return(exp(-((u/params[1])^2)*((sgn1*u + z1 - params[3])/sd.t) -
                 (sgn2*u + z2 - params[3])/sd.t)));
  }
  return(integrate(I, lower.limit, upper.limit, stop.on.error=1)$value);
}

sigma.int <- function(z, sgn, params, sd.t, lower.limit, upper.limit){
  I <- function(u){
    arg.in = sgn*u + z - params[3];
    return(exp(-((u/params[1])^2)*2*(u^2/params[1]^3)*pnorm(arg.in/sd.t) -
              params[1]*arg.in/(2*sd.t^3)*dnorm(arg.in/sd.t)));
  }
  return(integrate(I, lower.limit, upper.limit, stop.on.error=1)$value);
}

sigmab.int <- function(z, sgn, params, sd.t, lower.limit, upper.limit){
  I <- function(u){
    arg.in = sgn*u + z - params[3];
    return(exp(-((u/params[1])^2)*(-params[2]*arg.in*dnorm(arg.in/sd.t)/sd.t^3)));
  }
  return(integrate(I, lower.limit, upper.limit, stop.on.error=1)$value);
}

beta.i <- function(z, params, sd.t, sd.tb, lower.limit, upper.limit){
  return(params[1]*exp(-((z/sd.tb)^2)
                 *(pnorm((upper.limit*sd.tb - params[1]^2*z/sd.tb)/(params[1]*sd.t)) -
}

cfun1 <- function(theta, ratings.1, ratings.2, i.limits){
  n.class = nrow(i.limits);
  n = length(ratings.1);
  f = n*log(theta[1]);
  sdev.th = sqrt(0.5*theta[1]^2 + theta[2]^2);
  f.table <- mat.or.vec(n.class, n.class)
  for (lb1 in 1:n.class){
    for (lb2 in lb1:n.class){
      i1.lb <- i.limits[lb1,1]
      i1.ub <- i.limits[lb1,2]
      i2.lb <- i.limits[lb2,1]
      i2.ub <- i.limits[lb2,2]
      10
  }
intg <- f.int(i2.ub, 1, theta, sdev.th, 0.5*(i1.lb-i2.ub), 0.5*(i1.ub-i2.ub)) + f.int(i1.ub, -1, theta, sdev.th, 0.5*(i1.ub-i2.ub), 0.5*(i1.ub-i2.lb)) - f.int(i1.lb, -1, theta, sdev.th, 0.5*(i1.lb-i2.ub), 0.5*(i1.lb-i2.lb)) - f.int(i2.lb, 1, theta, sdev.th, 0.5*(i1.lb-i2.lb), 0.5*(i1.ub-i2.lb))

if(intg < 0){
  stop('f.table < 0')
} else{
  f.table[lb1, lb2] <- - log(intg)
}

f.table[lb2, lb1] <- f.table[lb1, lb2]

nonzero <- table(ratings.1, ratings.2) > 0
f = f + sum(table(ratings.1, ratings.2)[nonzero]*f.table[nonzero]);
return(f);

cfun2 <- function(theta, ratings.1, ratings.2, i.limits){
  n.class = nrow(i.limits);
  n = length(ratings.1);
  f = n*log(theta[1]);
  sdev.th = sqrt(0.5*theta[1]^2 + theta[2]^2);
  f.table <- mat.or.vec(n.class, n.class)
  for (lb1 in 1:n.class){
    for (lb2 in lb1:n.class){
      i1.lb <- i.limits[lb1,1]
      i1.ub <- i.limits[lb1,2]
      i2.lb <- i.limits[lb2,1]
      i2.ub <- i.limits[lb2,2]

      if((i1.ub - i2.ub)<(i1.lb - i2.lb)){
        intg <- fmod.int(i2.ub, 1, i1.lb, -1, theta, sdev.th, 0.5*(i1.ub-i2.ub), 0.5*(i1.lb-i2.ub)) + fmod.int(i1.ub, -1, i1.lb, -1, theta, sdev.th, 0.5*(i1.ub-i2.ub), 0.5*(i1.lb-i2.lb)) + fmod.int(i1.ub, -1, i2.ub, 1, theta, sdev.th, 0.5*(i1.lb-i2.lb), 0.5*(i1.ub-i2.lb))
      } else if((i1.ub - i2.ub)>(i1.lb - i2.lb)){
        intg <- fmod.int(i2.ub, 1, i1.lb, -1, theta, sdev.th, 0.5*(i1.ub-i2.ub), 0.5*(i1.lb-i2.ub)) + fmod.int(i2.ub, 1, i2.lb, 1, theta, sdev.th, 0.5*(i1.lb-i2.lb), 0.5*(i1.ub-i2.ub)) + fmod.int(i1.ub, -1, i2.lb, 1, theta, sdev.th, 0.5*(i1.lb-i2.lb), 0.5*(i1.ub-i2.ub))
      } else{
        intg <- fmod.int(i2.ub, 1, i1.lb, -1, theta, sdev.th, 0.5*(i1.ub-i2.ub), 0.5*(i1.lb-i2.ub)) + fmod.int(i1.ub, -1, i2.lb, 1, theta, sdev.th, 0.5*(i1.lb-i2.lb), 0.5*(i1.ub-i2.ub))
      }

      if(intg < 0){
        stop('f.table < 0')
      } else{
        f.table[lb1, lb2] <- - log(intg)
      }
    }
  }
  nonzero <- table(ratings.1, ratings.2) > 0
  f = f + sum(table(ratings.1, ratings.2)[nonzero]*f.table[nonzero]);
  return(f);
}
gradcfun1 <- function(theta, ratings.1, ratings.2, i.limits) {
  n.class = nrow(i.limits);
  n = length(ratings.1);
  Jf = c(n/theta[1], 0, 0);
  sdev.th = sqrt(0.5*theta[1]^2 + theta[2]^2);

  Jf1.table <- mat.or.vec(n.class, n.class)
  Jf2.table <- mat.or.vec(n.class, n.class)
  Jf3.table <- mat.or.vec(n.class, n.class)

  for (lb1 in 1:n.class) {
    for (lb2 in lb1:n.class) {
      i1.lb <- i.limits[lb1, 1]
      i1.ub <- i.limits[lb1, 2]
      i2.lb <- i.limits[lb2, 1]
      i2.ub <- i.limits[lb2, 2]

      corr.n <- (f.int(i2.ub, 1, theta, sdev.th, 0.5*(i1.lb-i2.ub), 0.5*(i1.ub-i2.ub)) +
                 f.int(i1.ub, -1, theta, sdev.th, 0.5*(i1.ub-i2.ub), 0.5*(i1.ub-i2.lb)) -
                 f.int(i1.lb, -1, theta, sdev.th, 0.5*(i1.lb-i2.ub), 0.5*(i1.lb-i2.lb)) -
                 f.int(i2.lb, 1, theta, sdev.th, 0.5*(i1.lb-i2.lb), 0.5*(i1.ub-i2.lb)));

      Jf1.table[lb1, lb2] <-
        -(sigma.int(i2.ub, 1, theta, sdev.th, 0.5*(i1.lb-i2.ub), 0.5*(i1.ub-i2.ub)) +
          sigma.int(i1.ub, -1, theta, sdev.th, 0.5*(i1.ub-i2.ub), 0.5*(i1.ub-i2.lb)) -
          sigma.int(i1.lb, -1, theta, sdev.th, 0.5*(i1.lb-i2.ub), 0.5*(i1.lb-i2.lb)) -
          sigma.int(i2.lb, 1, theta, sdev.th, 0.5*(i1.lb-i2.lb), 0.5*(i1.ub-i2.lb)))/
        corr.n;
      Jf1.table[lb2, lb1] <- Jf1.table[lb1, lb2]

      Jf2.table[lb1, lb2] <-
        -(sigmab.int(i2.ub, 1, theta, sdev.th, 0.5*(i1.lb-i2.ub), 0.5*(i1.ub-i2.ub)) +
          sigmab.int(i1.ub, -1, theta, sdev.th, 0.5*(i1.ub-i2.ub), 0.5*(i1.ub-i2.lb)) -
          sigmab.int(i1.lb, -1, theta, sdev.th, 0.5*(i1.lb-i2.ub), 0.5*(i1.lb-i2.lb)) -
          sigmab.int(i2.lb, 1, theta, sdev.th, 0.5*(i1.lb-i2.lb), 0.5*(i1.ub-i2.lb)))/
        corr.n;
      Jf2.table[lb2, lb1] <- Jf2.table[lb1, lb2]

      Jf3.table[lb1, lb2] <-
        (beta.i(theta[3]-i2.ub, theta, sdev.th, sdev.thb, 0.5*(i1.lb-i2.ub), 0.5*(i1.ub-i2.ub)) +
         beta.i(i1.ub-theta[3], theta, sdev.th, sdev.thb, 0.5*(i1.ub-i2.ub), 0.5*(i1.ub-i2.lb)) -
         beta.i(i1.lb-theta[3], theta, sdev.th, sdev.thb, 0.5*(i1.lb-i2.ub), 0.5*(i1.lb-i2.lb)) -
         beta.i(theta[3]-i2.lb, theta, sdev.th, sdev.thb, 0.5*(i1.lb-i2.lb), 0.5*(i1.ub-i2.lb)))/
        corr.n;
      Jf3.table[lb2, lb1] <- Jf3.table[lb1, lb2]
    }
  }
  nonzero <- table(ratings.1, ratings.2) > 0
  return(Jf);
gradcfun2 <- function(theta, ratings.1, ratings.2, i.limits)
  n.class = nrow(i.limits);
  n = length(ratings.1);
  Jf = c(n/theta[1], 0, 0);
  sdev.th = sqrt(0.5*theta[1]^2 + theta[2]^2);

  Jf1.table <- mat.or.vec(n.class, n.class)
  Jf2.table <- mat.or.vec(n.class, n.class)
  Jf3.table <- mat.or.vec(n.class, n.class)

  for (lb1 in 1:n.class){
    for (lb2 in lb1:n.class){
      i1.lb <- i.limits[lb1,1]
      i1.ub <- i.limits[lb1,2]
      i2.lb <- i.limits[lb2,1]
      i2.ub <- i.limits[lb2,2]

      if((i1.ub - i2.ub)<(i1.lb - i2.lb)){
        corr.n <- fmod.int(i2.ub,1,i1.lb,-1,theta,sdev.th,0.5*(i1.lb-i2.ub),0.5*(i1.ub-i2.ub)) +
                 fmod.int(i1.ub,-1,i1.lb,-1,theta,sdev.th,0.5*(i1.ub-i2.ub),0.5*(i1.ub-i2.ub)) +
                 fmod.int(i1.ub,-1,i2.lb,1,theta,sdev.th,0.5*(i1.lb-i2.lb),0.5*(i1.ub-i2.lb))
      }
      if((i1.ub - i2.ub)>(i1.lb - i2.lb)){
        corr.n <- fmod.int(i2.ub,1,i1.lb,-1,theta,sdev.th,0.5*(i1.lb-i2.ub),0.5*(i1.ub-i2.ub)) +
                 fmod.int(i1.ub,-1,i2.lb,1,theta,sdev.th,0.5*(i1.lb-i2.lb),0.5*(i1.ub-i2.lb)) +
                 fmod.int(i1.ub,-1,i2.ub,1,theta,sdev.th,0.5*(i1.ub-i2.ub),0.5*(i1.ub-i2.ub))
      }
      if((i1.ub - i2.ub)==(i1.lb - i2.lb)){
        corr.n <- fmod.int(i2.ub,1,i1.lb,-1,theta,sdev.th,0.5*(i1.lb-i2.ub),0.5*(i1.ub-i2.ub)) +
                 fmod.int(i1.ub,-1,i2.lb,1,theta,sdev.th,0.5*(i1.lb-i2.lb),0.5*(i1.ub-i2.lb))
      }

      Jf1.table[lb1, lb2] <-
      -(sigma.int(i2.ub,1,theta,sdev.th,0.5*(i1.lb-i2.ub),0.5*(i1.ub-i2.ub)) +
        sigma.int(i1.ub,-1,theta,sdev.th,0.5*(i1.ub-i2.ub),0.5*(i1.ub-i2.ub)) +
        sigma.int(i1.ub,-1,theta,sdev.th,0.5*(i1.ub-i2.ub),0.5*(i1.ub-i2.ub)) -
        sigma.int(i2.ub,1,theta,sdev.th,0.5*(i1.lb-i2.ub),0.5*(i1.ub-i2.ub)))/
        corr.n;

      Jf1.table[lb2, lb1] <- Jf1.table[lb1, lb2]

      Jf2.table[lb1, lb2] <-
      -(sigmab.int(i2.ub,1,theta,sdev.th,0.5*(i1.lb-i2.ub),0.5*(i1.ub-i2.ub)) +
        sigmab.int(i1.ub,-1,theta,sdev.th,0.5*(i1.ub-i2.ub),0.5*(i1.ub-i2.ub)) +
        sigmab.int(i1.ub,-1,theta,sdev.th,0.5*(i1.ub-i2.ub),0.5*(i1.ub-i2.ub)) -
        sigmab.int(i2.ub,1,theta,sdev.th,0.5*(i1.lb-i2.ub),0.5*(i1.ub-i2.ub)))/
        corr.n;

      Jf2.table[lb2, lb1] <- Jf2.table[lb1, lb2]

      Jf3.table[lb1, lb2] <-
      (beta.i(theta[3]-i2.ub,theta,sdev.th,sdev.thb,0.5*(i1.lb-i2.ub),0.5*(i1.ub-i2.ub)) +
       beta.i(i1.ub-theta[3],theta,sdev.th,sdev.thb,0.5*(i1.ub-i2.ub),0.5*(i1.ub-i2.ub)) -
       beta.i(i1.ub-theta[3],theta,sdev.th,sdev.thb,0.5*(i1.lb-i2.ub),0.5*(i1.ub-i2.ub)) -
       beta.i(theta[3]-i2.ub,theta,sdev.th,sdev.thb,0.5*(i1.lb-i2.ub),0.5*(i1.ub-i2.ub)))/
corr.n;
  Jf3.table[lb2, lb1] <- Jf3.table[lb1, lb2]
}

nonzero <- table(ratings.1, ratings.2) > 0
return(Jf);

intervalICC.est1 <-
function(ratings, classes, c.limits, theta0){
  costFunc <- function(theta){
    return(cfun1(theta, ratings$t1, ratings$t2, c.limits))
  }
  grad.costFunc <- function(theta){
    return(gradcfun1(theta, ratings$t1, ratings$t2, c.limits))
  }

  th_optim <- tryCatch({
    constrOptim(theta=theta0, f=costFunc, grad=grad.costFunc, 
    ui=rbind(c(1,0,0), c(0,1,0)), ci=c(.Machine$double.eps, .Machine$double.eps), 
    method = "BFGS"), 
    error = function(e){
      print("Numerical optimization failed; try optim.method=2"); return(0)}
  })

  sigma2.b <- th_optim$par[2]^2
  sigma2.w <- th_optim$par[1]^2
  mu <- th_optim$par[3]
  icc <- sigma2.b/(sigma2.b + sigma2.w)
  loglikelihood <- -(th_optim$value - nrow(ratings)*log(pi)/2)

  list(icc = icc,
       sigma2.b = sigma2.b,
       sigma2.w = sigma2.w,
       mu = mu,
       loglikelihood = loglikelihood)
}

intervalICC.est2 <-
function(ratings, classes, c.limits, theta0){
  costFunc <- function(theta){
    return(cfun2(theta, ratings$t1, ratings$t2, c.limits))
  }
  grad.costFunc <- function(theta){
    return(gradcfun2(theta, ratings$t1, ratings$t2, c.limits))
  }

  th_optim <- tryCatch({
    constrOptim(theta=theta0, f=costFunc, grad=grad.costFunc, 
    ui=rbind(c(1,0,0), c(0,1,0)), ci=c(.Machine$double.eps, .Machine$double.eps), 
    method = "BFGS"), 
    error = function(e){
      print("Numerical optimization failed; try optim.method=1"); return(0)}
  })
\[
\text{sigma2.b} \leftarrow \text{th_optim$par[2]}^2 \\
\text{sigma2.w} \leftarrow \text{th_optim$par[1]}^2 \\
\text{mu} \leftarrow \text{th_optim$par[3]} \\
\text{icc} \leftarrow \text{sigma2.b}/(\text{sigma2.b + sigma2.w}) \\
\text{loglikelihood} \leftarrow -\text{th_optim$value - nrow(ratings)*log(pi)/2}
\]

\[
\text{list(icc = icc,} \\
\quad \text{sigma2.b = sigma2.b,} \\
\quad \text{sigma2.w = sigma2.w,} \\
\quad \text{mu = mu,} \\
\quad \text{loglikelihood = loglikelihood)}
\]

\[
7.2 \text{ Simulation code}
\]

\[
n \leftarrow 1000; \quad \# \text{number of respondents (cases)}
\]

\[
n.\text{cat} \leftarrow 5; \quad \# \text{number of categories}
\]

\[
equal.\text{widths} \leftarrow \text{TRUE}; \quad \# \text{FALSE for unequal widths of categories}
\]

\[
n.\text{errc} \leftarrow 0; \quad \# \text{number of failure cases for ICC for uncensored data (ICC_c)}
\]

\[
n.\text{errd} \leftarrow 0; \quad \# \text{number of failure cases for midpoint method (ICC_MID)}
\]

\[
n.\text{erri} \leftarrow 0; \quad \# \text{number of failure cases for maximum likelihood estimator (ICC_MLE)}
\]

\[
n.\text{iter} \leftarrow 100; \quad \# \text{number of simulations}
\]

\[
\text{results} \leftarrow \text{mat.or.vec(n.iter,19)};
\]

\[
\text{icc.values} \leftarrow 0.7;
\]

\[
\text{for(icc in icc.values)}{
\quad n.\text{errc} = 0;
\quad n.\text{errd} = 0;
\quad n.\text{erri} = 0;
\}
\]

\[
k \leftarrow 1;
\]

\[
\text{while (k <= n.iter)}{
\quad \text{errors = FALSE;} \\
\quad \text{sigmab} \leftarrow \text{runif(1,0,50);} \quad \# \text{square-root of between-class variability}
\quad \text{sigmae} \leftarrow \text{sigmab*sqrt((1-icc)/icc);} \quad \# \text{square-root of within-class variability}
\quad \text{mu} \leftarrow 0; \quad \# \text{grand mean}
\]

\[
\text{data1} \leftarrow \text{mat.or.vec(n,1);} \quad \# \text{data for the first time point}
\]

\[
\text{data2} \leftarrow \text{mat.or.vec(n,1);} \quad \# \text{data for the second time point}
\]

\[
\# \text{generate continuous data}
\]

\[
\text{for (i in 1:n)}{
\quad \text{bi} \leftarrow \text{rnorm(n=1, mean=0, sd=sigmab)};
\quad \text{data1[i]} \leftarrow \text{mu + bi + rnorm(n=1, mean=0, sd=sigmae)};
\quad \text{data2[i]} \leftarrow \text{mu + bi + rnorm(n=1, mean=0, sd=sigmae)};
\}
\]

\[
\# \text{calculate ICC from continuous data (as a reference)}
\]

\[
\text{data.c} \leftarrow \text{as.data.frame(rbind(cbind(data1,1:n), cbind(data2,1:n))))}
\]

\[
\text{colnames(data.c)} \leftarrow \text{c("x","id")}
\]
modc <- tryCatch({lme(x ~ 1, random = ~1 | id, na.action = na.omit, data=data.c)},
    error = function(e) {print("Error: could not calculate ICC_c"); return(0)})

# partition data into n.cat categories
minf <- min(min(data1), min(data2));
maxf <- max(max(data1), max(data2));

if(equal.widths){
    cat.width <- (maxf - minf)/n.cat;
    inner.limits <- (1:(n.cat-1))*cat.width + minf;
} else{
    inner.limits <- sort(runif(n.cat-1,minf,maxf));
}

limits <- c(-Inf, minf, inner.limits, maxf, Inf); # cut-off points for categories
categs <- rep(0,n.cat)
for(ind in 1:n.cat){
    categs[ind] <- mean(limits[(ind+1):(ind+2)]); # categories as midpoints of cut-off points
}
data1.cat <- mat.or.vec(n,1);
data2.cat <- mat.or.vec(n,1);
for (i in 1:n){
    ind <- max(which(limits <= data1[i]));
data1.cat[i] <- categs[min(ind-1,n.cat)];
    ind <- max(which(limits <= data2[i]));
data2.cat[i] <- categs[min(ind-1,n.cat)];
}

# calculate ICC from grouped data; midpoint method
data.cat <- as.data.frame(rbind(cbind(data1.cat,1:n), cbind(data2.cat,1:n)))
colnames(data.cat) <- c("x","id")
modd <- tryCatch({lme(x ~ 1, random = ~1 | id, na.action = na.omit, data=data.cat)},
    error = function(e) {print("Error: could not calculate ICC_MID"); return(0)})

# calculate ICC from grouped data; maximum likelihood method
i.limits=cbind(limits[2:(n.cat+1)],limits[3:(n.cat+2)]);
ratings.1=factor(data1.cat, levels=categs); ratings.2=factor(data2.cat, levels=categs);
theta0 <- c( max(sd(apply(cbind(data1.cat,data2.cat),1,mean)), .Machine$double.eps+1e-10),
            max(mean(apply(cbind(data1.cat,data2.cat),1,sd)), .Machine$double.eps+1e-10),
            mean(data.cat$x))
cfun <- function(theta){return(cfun1(theta, ratings.1, ratings.2, i.limits))}
gradcfun <- function(theta){return(gradcfun1(theta, ratings.1, ratings.2, i.limits))}
# to optimize the log-likelihood given by (6), use:
# cfun <- function(theta){return(cfun2(theta, ratings.1, ratings.2, i.limits))}
# gradcfun <- function(theta){return(gradcfun2(theta, ratings.1, ratings.2, i.limits))}

th_optim <- tryCatch({constrOptim(theta=theta0, f=cfun, grad=gradcfun,
    ui=rbind(c(1,0,0),c(0,1,0)), ci=c(.Machine$double.eps,.Machine$double.eps),
    method = "BFGS")}, error = function(e){
    print("Error: could not calculate ICC_MLE"); return(0)})

# equivalent to:
# intervalICC(data1.cat, data2.cat, categs, i.limits, optim.method = 1)
# we use direct call to cost function cfun here for speed

# extract parameters from fitted models
if(class(modc)!="lme") {
    n.errc=n.errc+1;
    errors = TRUE;
} else {
    t0 <- as.numeric(VarCorr(modc)[1, 1])
    sig2 <- as.numeric(VarCorr(modc)[2, 1])
    icc_c <- t0/(t0 + sig2)
    sigmab_c <- sqrt(t0)
    sigmae_c <- sqrt(sig2)
    mu_c <- summary(modc)$tTable[1,1]}

if(class(modd)!="lme") {
    n.errd=n.errd+1;
    errors=TRUE;
} else {
    t0 <- as.numeric(VarCorr(modd)[1, 1])
    sig2 <- as.numeric(VarCorr(modd)[2, 1])
    icc_mid <- t0/(t0 + sig2)
    sigmab_mid <- sqrt(t0)
    sigmae_mid <- sqrt(sig2)
    mu_mid <- summary(modd)$tTable[1,1]}

if(class(th_optim)!="list") {
    n.errri=n.errri+1;
    errors=TRUE;
} else {
    sigmab_mle <- th_optim$par[2]
    sigmae_mle <- th_optim$par[1]
    mu_mle <- th_optim$par[3]}

if(errors==FALSE){
    results[k,] = c(mu, sigmab, sigmae, icc, icc_c, icc_mid, icc_mle,
            mu_c, sigmab_c, sigmae_c, mu_mid, sigmab_mid, sigmae_mid,
            mu_mle, sigmab_mle, sigmae_mle, n.errc, n.errd, n.errri);
    k = k+1;
}
}
colnames(results) <- c("mu", "sigmab", "sigmae", "icc", "icc_c", "icc_mid", "icc_mle",
                "mu_c", "sigmab_c", "sigmae_c", "mu_mid", "sigmab_mid", "sigmae_mid",
                "mu_mle", "sigmab_mle", "sigmae_mle", "errors_c", "errors_mid", "errors_mle")
7.3 Using the R code to estimate ICC from grouped data

In this subsection we work through one example of grouped data, using the code provided in 7.1. Prior to running the code given below, all functions from 7.1 need to be loaded into the R workspace.

For each predefined category in questionnaire, we first need to specify category label and cut-off points, e.g., for the question “How many cigarettes/day do you smoke” (from the Fagerström Test for Nicotine Dependence⁴):

<table>
<thead>
<tr>
<th>Category</th>
<th>Label</th>
<th>Lower cut-off point</th>
<th>Upper cut-off point</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 or less</td>
<td>0</td>
<td>0</td>
<td>10.5</td>
</tr>
<tr>
<td>11 - 20</td>
<td>1</td>
<td>10.5</td>
<td>20.5</td>
</tr>
<tr>
<td>21 - 30</td>
<td>2</td>
<td>20.5</td>
<td>30.5</td>
</tr>
<tr>
<td>31 and more</td>
<td>3</td>
<td>30.5</td>
<td>40</td>
</tr>
</tbody>
</table>

In the main function intervalICC, category labels correspond to the argument classes, while cut-off points correspond to c.limits. In R, we would type:

```r
classes <- 0:3
c.limits <- matrix(c(0,10.5,10.5,20.5,20.5,30.5,30.5,40), byrow=TRUE, nrow=4)
```

Now we specify questionnaire data. If we have N = 10 respondents, we need to specify two vectors of length N consisting of category labels, e.g.,

```r
q1 <- c(0,0,3,2,1,0,1,2,1,0)
q2 <- c(0,1,2,3,1,0,0,2,1,0)
```

In our example, this would mean that the first respondent answered that he smoked 10 or less cigarettes in both questionnaires. The second respondent selected the category “10 or less” in the first questionnaire and the category “11-20” in the second one, and so on. Finally, calling

```r
intervalICC(ratings1=q1, ratings2=q2, classes=classes, c.limits=c.limits)
```

gives us ICC of 0.87. In the case that the data require transforming (such as adding a constant and applying natural logarithm, which would be appropriate in this example), the transformation needs to be applied only to c.limits.

For information on estimating ICC from grouped data using the package iRepro, please visit [http://www.imi.hr/~jkovacic/irepro.html](http://www.imi.hr/~jkovacic/irepro.html).

References


