Web-based Supplementary Materials for "Linear Calibration with Grouped Data" by Brandon M. Greenwell and Christine M. Schubert

Web Appendix A

The ML Estimator of x_0 for the Random Intercept Model

In this appendix, we show that the inverse estimator, $\widehat{x}_0 = \mu^{-1}\left(\mathcal{Y}_0; \widehat{\boldsymbol{\beta}}\right)$, is the ML estimator of x_0 for the random intercept model. For this model, note that the scaled variance-covariance matrix of the random effect becomes $\boldsymbol{G}^{\dagger} = \tau \boldsymbol{I}$ and that $\boldsymbol{Z}_i = \mathbf{1}_i$ (a column vector of all ones), hence, $\boldsymbol{V}_i = \sigma_{\epsilon}^2 \left(\boldsymbol{I}_i + \tau \boldsymbol{J}_i\right)$ where $\boldsymbol{J}_i = \mathbf{1}_i \mathbf{1}_i'$. Therefore, we can write

$$\boldsymbol{\mathcal{Y}}_i \sim \mathcal{N}\left\{\boldsymbol{X}_i \boldsymbol{\beta}, \sigma_{\epsilon}^2 \left(\boldsymbol{I}_i + \tau \boldsymbol{J}_i\right)\right\}, \quad i = 1, \ldots, m,$$

where $\boldsymbol{X}_i$ is an $n_i \times 2$ design matrix with j-th row equal to $\boldsymbol{X}_{ij}' = (1, x_{ij})$, $\boldsymbol{\beta} = (\beta_0, \beta_1)'$ is a vector of fixed effects, σ_{ϵ}^2 is the within-subject variance, and $\sigma_{\epsilon}^2 \tau$ is the variance of the random intercepts. Ignoring constants, the log-likelihood for the data is

$$\begin{aligned} \ell_{\mathrm{I}}\left(\boldsymbol{\beta}, \sigma_{\epsilon}^2, \tau\right) = -\frac{N}{2} \log\left(\sigma_{\epsilon}^2\right) - \frac{1}{2} \sum_{i=1}^{m} \log \left|\boldsymbol{I} + \tau \boldsymbol{J}_i\right| \\ - \frac{1}{2\sigma_{\epsilon}^2} \sum_{i=1}^{m} \left(\boldsymbol{\mathcal{Y}}_i - \boldsymbol{X}_i \boldsymbol{\beta}\right)' \left(\boldsymbol{I} + \tau \boldsymbol{J}_i\right)^{-1} \left(\boldsymbol{\mathcal{Y}}_i - \boldsymbol{X}_i \boldsymbol{\beta}\right). \end{aligned}$$

The subscript "I" is there to remind us that this is the likelihood for the data from the first stage of the calibration experiment. Using the following formulas (Demidenko, 2013, pg. 49),

- $\left|\boldsymbol{I} + \tau \boldsymbol{J}_i\right| = 1 + n_i \tau$;
- $\left(\boldsymbol{I} + \tau \boldsymbol{J}_i\right)^{-1} = \boldsymbol{I} - \frac{\tau}{1 + n_i \tau} \boldsymbol{J}_i$;

the log-likelihood simplifies to

$$\ell_{\mathrm{I}}\left(\boldsymbol{\beta}, \sigma_\epsilon^2, \tau\right) = -\frac{N}{2}\log\left(\sigma_\epsilon^2\right) - \frac{1}{2}\sum_{i=1}^{m}\log\left(1+n_i\tau\right) - \frac{1}{2\sigma_\epsilon^2}\sum_{i=1}^{m}\left(\boldsymbol{\mathcal{Y}}_i - \boldsymbol{X}_i\boldsymbol{\beta}\right)'\left(\boldsymbol{I} - \frac{\tau}{1+n_i\tau}\boldsymbol{J}_i\right)\left(\boldsymbol{\mathcal{Y}}_i - \boldsymbol{X}_i\boldsymbol{\beta}\right).$$

Similarly, the log-likelihood for $\mathcal{Y}_0$ (i.e., the data from the second stage of the calibration experiment) is

$$\ell_{\mathrm{II}}\left(\boldsymbol{\beta}, \sigma_\epsilon^2, \tau, x_0\right) = -\frac{1}{2}\log\left(\sigma_\epsilon^2\right) - \frac{1}{2}\log\left(1+\tau\right) - \frac{1}{2\sigma_\epsilon^2\left(1+\tau\right)}\left(\mathcal{Y}_0 - \beta_0 - \beta_1 x_0\right)^2.$$

From the independence of $\boldsymbol{\mathcal{Y}}$ and $\mathcal{Y}_0$, the log-likelihood for the pooled data, denoted $\ell\left(\boldsymbol{\beta}, \sigma_\epsilon^2, \tau, x_0\right)$, is given by

$$\begin{aligned}\ell\left(\boldsymbol{\beta}, \sigma_\epsilon^2, \tau, x_0\right) &= \ell_{\mathrm{I}}\left(\boldsymbol{\beta}, \sigma_\epsilon^2, \tau\right) + \ell_{\mathrm{II}}\left(\boldsymbol{\beta}, \sigma_\epsilon^2, \tau, x_0\right) \\ &= -\frac{N+1}{2}\log\left(\sigma_\epsilon^2\right) - \frac{1}{2}\sum_{i=1}^{m}\log\left(1+n_i\tau\right) - \frac{1}{2}\log\left(1+\tau\right) \\ &\quad - \frac{1}{2\sigma_\epsilon^2}\sum_{i=1}^{m}\left(\boldsymbol{\mathcal{Y}}_i - \boldsymbol{X}_i\boldsymbol{\beta}\right)'\left(\boldsymbol{I} - \frac{\tau}{1+n_i\tau}\boldsymbol{J}_i\right)\left(\boldsymbol{\mathcal{Y}}_i - \boldsymbol{X}_i\boldsymbol{\beta}\right) \\ &\quad - \frac{1}{2\sigma_\epsilon^2\left(1+\tau\right)}\left(\mathcal{Y}_0 - \beta_0 - \beta_1 x_0\right)^2.\end{aligned}$$

Thus, the full log-likelihood is the sum of two parts, the log-likelihood for the standards, and the log-likelihood for the unknown. Equating to zero the partial derivative of the full log-likelihood with respect to the parameter x_0 results in $\widetilde{x}_0\left(\boldsymbol{\beta}\right) = \left(\mathcal{Y}_0 - \beta_0\right)/\beta_1$. In other words, for any value of $\boldsymbol{\beta}$, $\widetilde{x}_0\left(\boldsymbol{\beta}\right)$ maximizes the likelihood with respect to x_0. Substituting this into the log-likelihood yields the *profiled log-likelihood*

$$\ell_p\left(\boldsymbol{\beta}, \sigma_\epsilon^2, \tau\right) = -\frac{N+1}{2}\log\left(\sigma_\epsilon^2\right) - \frac{1}{2}\sum_{i=1}^{m}\log\left(1+n_i\tau\right) - \frac{1}{2}\log\left(1+\tau\right) - \frac{1}{2\sigma_\epsilon^2}\sum_{i=1}^{m}\left(\boldsymbol{\mathcal{Y}}_i - \boldsymbol{X}_i\boldsymbol{\beta}\right)'\left(\boldsymbol{I} - \frac{\tau}{1+n_i\tau}\boldsymbol{J}_i\right)\left(\boldsymbol{\mathcal{Y}}_i - \boldsymbol{X}_i\boldsymbol{\beta}\right),$$

Similarly, equating the partial derivative of $\ell_p\left(\boldsymbol{\beta}, \sigma_\epsilon^2, \tau\right)$, with respect to the parameter σ_ϵ^2, to zero yields

$$\widetilde{\sigma}_\epsilon^2\left(\boldsymbol{\beta}, \tau\right) = \frac{1}{N+1}\sum_{i=1}^{m}\left(\boldsymbol{\mathcal{Y}}_i - \boldsymbol{X}_i\boldsymbol{\beta}\right)\left(\boldsymbol{I} - \frac{\tau}{1+n_i\tau}\boldsymbol{J}_i\right)\left(\boldsymbol{\mathcal{Y}}_i - \boldsymbol{X}_i\boldsymbol{\beta}\right)'.$$

Substituting this into the profiled log-likelihood and simplifying we get

$$\ell_p(\boldsymbol{\beta}, \tau) = -\frac{N+1}{2} \log \left\{ \sum_{i=1}^{m} (\boldsymbol{\mathcal{Y}}_i - \boldsymbol{X}_i\boldsymbol{\beta})' \left(\boldsymbol{I} - \frac{\tau}{1+n_i\tau} \boldsymbol{J}_i \right) (\boldsymbol{\mathcal{Y}}_i - \boldsymbol{X}_i\boldsymbol{\beta}) \right\} - \frac{1}{2} \sum_{i=1}^{m} \log(1+n_i\tau) - \frac{1}{2} \log(1+\tau),$$

The parameters x_0 and σ_ϵ^2 have been "profiled out", resulting in a simpler log-likelihood in only $p+1$ parameters. We could continue in this fashion with the parameter $\boldsymbol{\beta}$ as well, although, it is quite easy to see that the value of $\boldsymbol{\beta}$ that maximizes $\ell_p(\boldsymbol{\beta}, \tau)$ is just the usual generalized least squares estimator, $\widetilde{\boldsymbol{\beta}}(\tau)$. Furthermore, it can be shown (Demidenko, 2013) that, for the balanced case, $\widetilde{\boldsymbol{\beta}}(\tau) = \widetilde{\boldsymbol{\beta}}$ does not depend on τ and in fact reduces to the ordinary least squares estimator $\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{\mathcal{Y}}$. Thus, the ML estimator of x_0 for the balanced random intercept model is simply

$$\widehat{x}_0 = \widetilde{x}_0\left(\widehat{\boldsymbol{\beta}}\right) = \frac{\mathcal{Y}_0 - \widehat{\beta}_0}{\widehat{\beta}_1},$$

where

$$\widehat{\beta}_1 = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m}(x_{ij}-\bar{x})(y_{ij}-\bar{y})}{\sum_{i=1}^{n}\sum_{j=1}^{m}(x_{ij}-\bar{x})^2}, \quad \widehat{\beta}_0 = \bar{y} - \widehat{\beta}_1\bar{x}.$$

Web Appendix B

Example R code

The following snippets of R code are provided as a template for implementing the various methods outlined in our paper. The first example loads the necessary packages and data, and fits the LMM discussed in the paper.

R Example 1

```
## Load packages
library(car)  # for deltaMethod() function
library(nlme)  # for fitting LMMs
library(lme4)  # for fitting LMMs and parametric bootstrap
library(boot)  # for calculating bootstrap CI's


## Bladder data
subject <- rep(1:23, times = 8)
volume <- rep(c(10, 25, 50, 75, 100, 125, 150, 175),
    each = 23)
HD <- c(13.2, 11.1, 10.3, NA, 4.8, 7.7, NA, 5.9, 1.9,
```

```
    6.5, 19.8, 14.6, NA, NA, 9.7, 17.2, 10.6, 19.3,
    8.5, 6.9, 8.1, 14.8, 13.7, 27.4, 27.5, 15, 10,
    18.6, 12.6, 24, 28.4, 12.5, 16.7, 29.6, 27.1, 14,
    18.7, 20.3, 35.8, 23.6, 37.4, 31.3, 23.7, 22, 34.3,
    28.5, 41.6, 58.1, 34.2, 28.8, 29.9, 31.4, 46.9,
    44.4, 26.8, 30.6, 51.7, 49.8, 19.1, 35.8, 38.9,
    41.4, 49.9, 58.6, 54.8, 44, 39.1, 58.5, 41.5, 60.1,
    78.8, 49.4, 46.4, 39.4, 45.3, 50.4, 70.7, 54.4,
    41.8, 72.2, 67.5, 39.2, 49.6, 65.1, 69.7, 67.7,
    73.7, 78.3, 65.7, 44.7, 72.1, 59.8, 73.9, 91.5,
    71.3, 54.8, NA, 48, 67.8, 89.4, 63.1, 49.6, 81.9,
    79.1, 48.7, 65.6, 65.1, 81.9, 87.7, 79.4, 93, 80.3,
    68.9, 90.9, 77.5, 85.5, 98.3, 81.3, 69.4, NA, 66.6,
    81, 105.8, 83.5, 60.8, 95.1, 95.1, 67, 85.3, 86.9,
    96.6, 89.3, 102.6, NA, 93.6, 93.3, 105, 92.9, 95.6,
    111.4, 94, 73.9, NA, NA, 91.2, 113.5, 114.5, 80.1,
    115.4, 109.8, 72.7, 90.4, 98.6, 115, 108, 110.9,
    NA, 99.2, 102.4, 117.5, 99.4, 107.4, 121, 104.3,
    NA, NA, NA, 99.8, 127.3, 124, 87.1, NA, NA, NA,
    NA, 107.2, 117, 114.8, 122.4, NA, 112.2, 104.7,
    124.2, 113)
bladder <- data.frame(subject = subject, HD = HD, volume = volume)
bladder <- na.omit(bladder)

## Fit model
bladder.nlme <- lme(HD ~ volume + I(volume^2), data = bladder,
    random = list(subject = pdDiag(~volume)))
```

To obtain the inverse estimate, we find it useful to write a simple function. If the solution can not easily be written in closed-form, then the user can easily amend the following example to call the built-in R function `uniroot` to solve the equation

$$\mu\left(x_0; \widehat{\boldsymbol{\beta}}\right) - y_0 = 0$$

for x_0 numerically.

R Example 2

```
xest <- function(object, y0) {
    b <- unname(fixef(object))
    (-b[2] + sqrt(b[2]^2 - 4 * b[3] * (b[1] - y0)))/(2 *
        b[3])
}
(x0.est <- xest(bladder.nlme, y0 = 85))

## [1] 120.5
```

Obtaining the Wald-based interval is straightforward using the car package (Fox and Weisberg, 2011), as illustrated in the following snippet of code.

R Example 3

```
b <- unname(fixef(bladder.nlme))
var.y0 <- getVarCov(bladder.nlme)[1, 1] + x0.est^2 *
    getVarCov(bladder.nlme)[2, 2] + summary(bladder.nlme)$sigma^2
covmat <- diag(4)
covmat[1:3, 1:3] <- vcov(bladder.nlme)
covmat[4, 4] <- var.y0
params <- c(b0 = b[1], b1 = b[2], b2 = b[3], Y0 = 85)
g <- "(-b1 + sqrt(b1^2 - 4*b2*(b0-Y0))) / (2*b2)"
dm <- deltaMethod(params, g = g, vcov. = covmat)

## Approximate standard error
dm$SE

## [1] 26.2

## The Wald-based interval
(wald.ci <- x0.est + qnorm(c(0.025, 0.975)) * dm$SE)

## [1]  69.18 171.88
```

As discussed in our paper, the inversion interval is a little trickier since we need to write a new prediction function that returns an approximate standard error for the fitted values. We then invert an approximate prediction interval to obtain a confidence interval for x_0 as in the following snippet of code. As discussed in our paper, this interval may not exist, in which case the following code will produce in error. In our simulations, we augmented the `uniroot` function to return $\pm\infty$ in such cases.

R Example 4

```
predFun <- function(x) {
    z <- list(volume = x)
    fit <- predict(bladder.nlme, newdata = z, level = 0)
    se.fit <- sqrt(diag(cbind(1, unlist(z), unlist(z)^2) %*%
        bladder.nlme$varFix %*% t(cbind(1, unlist(z),
        unlist(z)^2))))
    list(fit = fit, se.fit = se.fit)
}
bounds <- function(x, w) {
    z <- list(volume = x)
    pred <- predFun(x)
    (85 - pred$fit)/sqrt(var.y0 + pred$se.fit^2) -
```

```
        w
}
lower <- uniroot(bounds, interval = c(min(bladder$volume),
    x0.est), w = qnorm(0.975), tol = 1e-10, maxiter = 1000)$root
upper <- uniroot(bounds, interval = c(x0.est, 250),
    w = qnorm(0.025), tol = 1e-10, maxiter = 1000)$root

## The inversion interval
(inversion.ci <- c(lower, upper))

## [1]  75.39 184.70
```

The last example shows how to use the new `bootMer` function from the well-known lme4 (Bates et al., 2014) package to implement the parametric bootstrap algorithm for calibration outlined in our paper. To obtain this confidence interval, we need to refit the model using the `lmer` function in lme4, this may produce a warning message for these data, but the model is well-defined and matches the output obtained from fitting the model using the base package nlme (Pinheiro et al., 2013).

R Example 5

```
## Refit model using lme4 package (may get a warning
## message)
bladder.lme4 <- lmer(HD ~ volume + I(volume^2) + (0 +
    1 | subject) + (0 + volume | subject), data = bladder)

## Warning:  Model failed to converge with max|grad| = 0.0321661 (tol
= 0.002)

## Calculate variance of Y0
var.y0 <- VarCorr(bladder.lme4)[[1]][1] + x0.est^2 *
    VarCorr(bladder.lme4)[[2]][1] + sigma(bladder.lme4)^2

## Bootstrap function
bootFun <- function(.) {

    ## Bootstrap inverse estimate
    if (all(getME(., "y") == bladder$HD)) {
        y0.boot <- 85
    } else {
        y0.boot <- 85 + rnorm(1, mean = 0, sd = sqrt(var.y0))
    }
    x0.boot <- xest(., y0 = y0.boot)

    ## Bootstrap predictive pivot
    mu0.boot <- as.numeric(crossprod(fixef(.), c(1,
```

```
        x0.est, x0.est^2)))
    var.y0.boot <- VarCorr(.)[[1]][1] + x0.est^2 *
        VarCorr(.)[[2]][1] + sigma(.)^2
    var.mu0.boot <- t(c(1, x0.est, x0.est^2)) %*% as.matrix(vcov(.)) %*%
        c(1, x0.est, x0.est^2)
    Q.boot <- (y0.boot - mu0.boot)/sqrt(var.y0.boot +
        var.mu0.boot)

    ## Return bootstrap estimates
    c(x0.boot, Q.boot)

}

## Run bootstrap simulation (will take a few
## minutes!)
set.seed(101)
bladder.pb <- bootMer(bladder.lme4, FUN = bootFun,
    nsim = 9999, parallel = "multicore", ncpus = 4)

## Get bootstrap Wald and percentile intervals for
## x0
boot.ci(bladder.pb, index = 1, type = c("norm", "perc"))

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 9991 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bladder.pb, type = c("norm", "perc"), index = 1)
##
## Intervals :
## Level      Normal             Percentile
## 95%   ( 64.3, 172.3 )   ( 75.4, 183.1 )
## Calculations and Intervals on Original Scale

## Bootstrap adjusted inversion interval
qstar <- quantile(bladder.pb$t[, 2], c(0.025, 0.975))
lower <- uniroot(bounds, interval = c(min(bladder$volume),
    x0.est), w = qstar[2], tol = 1e-10, maxiter = 1000)$root
upper <- uniroot(bounds, interval = c(x0.est, 250),
    w = qstar[1], tol = 1e-10, maxiter = 1000)$root
(inversion2.ci <- c(lower, upper))

## [1]  74.38 187.12
```

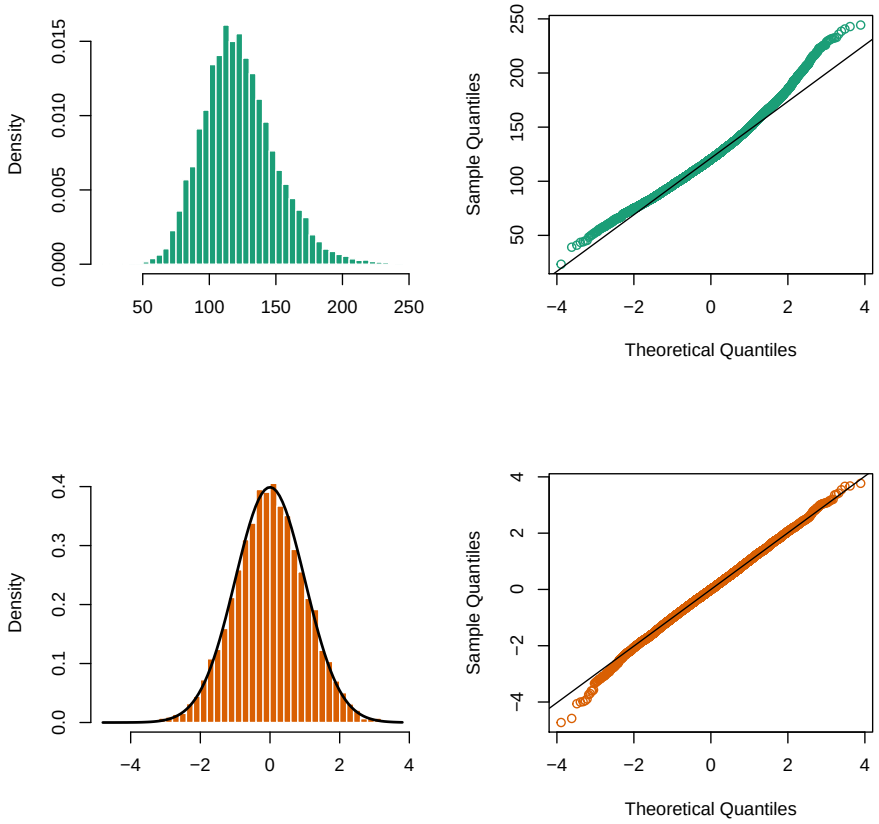

Figure 1: Histograms and normal Q-Q plots for the 9,999 bootstrap replicates of $\widehat{x}_0$ and $\mathcal{Q}$. *Bottom*: Inverse estimator, $\widehat{x}_0$. *Bottom*: Predictive pivot, $\mathcal{Q}$.

References

Bates, D., Maechler, M., Bolker, B., and Walker, S. (2014). *lme4: Linear mixed-effects models using Eigen and S4*. R package version 1.1-5.

Demidenko, E. (2013). *Mixed Models: Theory and Applications with R*. Wiley.

Fox, J. and Weisberg, S. (2011). *An R Companion to Applied Regression*. Sage, Thousand Oaks CA, second edition. R package version 2.0-19.

Pinheiro, J., Bates, D., DebRoy, S., Sarkar, D., and R Core Team (2013). *nlme: Linear and Nonlinear Mixed Effects Models*. R package version 3.1-110.