When a **multilevel** model includes either a non-linear transformation (such as the log-transformation) of the response variable, or of the expectations via a GLM link-function, then the interpretation of the results will be different compared to a standard Gaussian multilevel model; specifically, the estimates will be on a transformed scale and not in the original units, and the effects will no longer refer to the average effect in the population, instead they are conditional/cluster-specific. In this post, I will deal with linear mixed-effects models (LMM) that use a log-transformed outcome variable. This will be the first part of a three-part tutorial on some of the finer details of (G)LMMs, and how Bayes can make your (frequentist) life easier. In each post I will focus on:

* **log-transformation** of the dependent variable in a **multilevel** model
* Calculating both **multiplicative effects** (% change) and differences on the **untransformed scale**.
* **Subject-specific**/cluster-specific versus **population-average effects** (conditional versus marginal effects).
* **Intraclass correlations** on the transformed/link/latent scale or on the response/data/original scale.

In part 2 I will cover a GLMM with a binary outcome, and part 3 will focus on semicontinuous (hurdle/two-part) models when the outcome is a skewed continuous variable that include zeros.

The aim of this post is also to show:

* How simulation-based approaches like MCMC make it much easier to make inferences about transformed parameters.
* How the estimates from a multilevel model can be transformed to answer the same questions as population-average models or fixed effects models.

**Data-generating model: a lognormal LMM**

As an example, we will use a simple hierarchical design with clusters nested within either a treatment or a control condition. This is a classic 2-level multilevel model.

$$ \text{log}(y\_{ij} \mid u\_i) \sim \mathcal{N}(\beta\_0 + \beta\_1 TX\_{i} + u\_i, \sigma\_e^2) $$

$$ u\_i \sim \mathcal{N}(0, \sigma^2\_u) $$

In this model, the natural logarithm of \(y\) is normally distributed conditional on the cluster-specific effect \(u\_i\). Let’s assume \(y\) is the average money (in USD) lost gambling per week (since I do research on gambling disorder). Gambling expenditure is sometimes quite well described by a lognormal distribution, or a (generalized) Gamma distribution. Importantly, the methods in this post can quite easily be tweaked to work with other response distributions. Each treatment arm has \(n\_2\) clusters, and each cluster has \(n\_1\) observations. In this example clusters refer to e.g. schools, centers, groups, and the first level are subjects. However, it would be easy to use the contents of this post for a longitudinal model where clusters refer to subjects, and the subjects have repeated observations.

**Simulate data**

Let’s create a simple function that generates data from this model

#' Generate lognormal data with a random intercept

#'

#' @param n1 patients per cluster

#' @param n2 clusters per treatment

#' @param B0 log intercept

#' @param B1 log treatment effect

#' @param sd\_log log sd

#' @param u0 SD of log intercepts (random intercept)

#'

#' @return a data.frame

gen\_data <- function(n1, n2, B0, B1, sd\_log, u0) {

cluster <- rep(1:(2 \* n2), each = n1)

TX <- rep(c(0, 1), each = n1 \* n2)

u0 <- rnorm(2 \* n2, sd = u0)[cluster]

mulog <- (B0 + B1 \* TX + u0)

y <- rlnorm(2 \* n1 \* n2, meanlog = mulog, sdlog = sd\_log)

d <- data.frame(cluster,

TX,

y)

d

}

The model parameters are

set.seed(4445)

pars <- list("n1" = 30, # observations per cluster

"n2" = 10, # clusters per treatment

"B0" = log(500),

"B1" = log(0.5),

"sd\_log" = 0.5,

"u0" = 0.5)

d <- do.call(gen\_data,

pars)

**Fit the model**

We can fit this model using brms (or rstanarm, lme4, etc).

fit <- brm(y ~ 1 + TX + (1 | cluster),

family = lognormal(),

data = d,

cores = 4,

seed = 4445)

fit

## Family: lognormal

## Links: mu = identity; sigma = identity

## Formula: y ~ 1 + TX + (1 | cluster)

## Data: d (Number of observations: 600)

## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;

## total post-warmup samples = 4000

##

## Group-Level Effects:

## ~cluster (Number of levels: 20)

## Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat

## sd(Intercept) 0.64 0.12 0.45 0.91 640 1.01

##

## Population-Level Effects:

## Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat

## Intercept 6.21 0.21 5.80 6.61 583 1.00

## TX -0.70 0.30 -1.25 -0.09 548 1.00

##

## Family Specific Parameters:

## Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat

## sigma 0.51 0.01 0.48 0.54 1834 1.00

##

## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample

## is a crude measure of effective sample size, and Rhat is the potential

## scale reduction factor on split chains (at convergence, Rhat = 1).

Using the notation of our 2-level model, we get these results

| **Parameter** | **Estimate** |
| --- | --- |
| \(\hat\beta\_0\) | 6.21 |
| \(\hat\beta\_1\) | -0.70 |
| \(\hat\sigma\_u\) | 0.64 |
| \(\hat\sigma\_e\) | 0.51 |

**Interpreting the parameters**

All of the estimates are on the log scale, and they do not have an immediate interpretation on the natural scale. To further complicate things we should also remind ourselves that the effects from (G)LMM have cluster-specific interpretations (“subject-specific” in a longitudinal model when clusters == subjects). Researchers who work mostly with LMMs sometimes tend to forget this, since for LMMs the cluster-specific effects are equal to the population-average (marginal) effects. The non-linear transformation of either the data (like the log-transformation) or via the link function in (G)LMMs both means that back-transformation of the model’s estimates from e.g. the log scale does not give the expected values on the natural scale (cf. Jensen’s inequality).

In the figure below I try to illustrate both the cluster-specific nature of the model, and the transformation between the log scale and the natural data scale, without being too technical.

**How to calculate marginal effects on the data scale**

So how do we derive the numbers presented in the figure above? Well, for the lognormal distribution with the log mean \(\mu\) and the log SD \(\sigma\_e\) we have:

* mean: \(\text{exp}(\mu + \frac{\sigma\_e^2}{2})\)
* median: \(\text{exp}(\mu)\)

If we want to get the marginal mean we need to average this back-transformation over the random effects distribution. Since, as noted earlier, simply back-transforming the log coefficient (the expected mean on the log scale) will not give the expected mean on the original scale (money lost).

$$ \text{exp}(\beta\_0 + \frac{\sigma^2\_e}{2}) \ne \text{E} (Y\_{ij} \mid TX = 0)$$

The correct expectation is

$$ \text{E}(y\_{ij} \mid TX = 0) = \int \text{exp}(x + \sigma^2\_e/2) f\_{\mathcal{N}}(x, \beta\_{0}, \sigma\_{u}^2)dx$$

This integral is easily solved using numerical integration.

mu <- integrate(

f = function(x) {

exp(x + sd\_log ^ 2 / 2) \* dnorm(x, B0, sd = u0)

},

lower = B0 - 10 \* u0,

upper = B0 + 10 \* u0

)$value

mu

## [1] 642.0127

In this case, we can also derive an exact and simple solution using the moment-generating function of the lognormal distribution (see Proof 1: left as an exercise to the reader )

exp(B0 + (u0^2 + sd\_log^2)/2)

## [1] 642.0127

A third option is Monte Carlo integration. Were we simply simulate a large number of cluster outcomes on the log scale, back-transform to the natural scale, and then get the average. For multivariate integrals (e.g. models with correlated random effects) this is often the simplest solution that also works well in practice, although the package cubature offers a fast solution using adaptive numerical integration.

u\_i <- rnorm(1e5, 0, u0)

y <- exp(B0 + u\_i + sd\_log^2/2)

mean(y)

## [1] 643.239

We see that all three methods give the same result (within Monte Carlo error).

**Can we interpret the cluster-specific values?**

So, if the expected value for the outcome group is 642 USD, and \(\text{exp}(\beta\_0 + \frac{\sigma^2\_e}{2})\) = 567 USD is there a meaningful interpretation of 567? As shown in the figure above, this is the expected value for the cluster at the center of the distribution of cluster means on the *log scale*. Since percentiles are invariant under a monotonic transformation (and the mean is equal to the median for the normal distribution), this cluster (with \(u\_i = 0\)) is also at the median on the natural scale.

exp(B0 + sd\_log^2/2)

## [1] 566.5742

So to summarize, the cluster-specific effects are,

$$ \text{E}(y\_{ij} \mid u\_i, TX\_i) = \text{exp}(\beta\_0 + \beta\_1TX\_i + u\_i + \frac{\sigma^2\_e}{2}),$$

and the expected marginal mean in each treatment group is,

$$ \text{E}(y\_{ij} | TX\_i) = \text{exp}(\beta\_0 + \beta\_1TX\_i + \frac{\sigma^2\_u}{2} + \frac{\sigma^2\_e}{2})$$

**Average treatment effects on the data scale**

Using the results above, the cluster-specific average treatment effect on the original scale is,

$$ \text{E}(Y\_{ij} \mid u\_i, \text{TX} = 1) – \text{E}(Y\_{ij} \mid u\_i, \text{TX} = 0) = \\\  
\text{exp}(\beta\_0 + \beta\_1 + u\_i + \frac{\sigma^2\_e}{2}) – \text{exp}(\beta\_0 + u\_i + \frac{\sigma^2\_e}{2})$$

which is (for \(u\_i = 0\)),

## [1] -283.2871

And the marginal treatment effect on the original scale is,

$$ \text{E}(Y\_{ij} \mid \text{TX} = 1) – \text{E}(Y\_{ij} \mid \text{TX} = 0) =  
\text{exp}(\beta\_0 +\beta\_1 + \frac{\sigma^2\_e + \sigma^2\_u}{2}) – \text{exp}(\beta\_0 + \frac{\sigma^2\_e + \sigma^2\_u}{2})$$

## [1] -321.0064

**Multiplicate effects (% change)**

Most know that if we take the exponent of the log differences we get the % change. Generally, these multiplicative effects are also cluster-specific. However, for our simple model \(\text{exp}(\beta\_1)\) gives both the cluster-specific and population-average effect. We can see this if we look at the ratio of the expected conditional outcome in the treatment and control group,

$$  
\begin{align\*}  
\frac{\text{E}(Y\_{ij} \mid u\_i, \text{TX} = 1)}{\text{E}(Y\_{ij} \mid u\_i, \text{TX} = 0)} &= \frac{\text{exp}(\beta\_0 + \beta\_1 +u\_i + \frac{\sigma^2\_e}{2})}  
{\text{exp}(\beta\_0 + u\_i + \frac{\sigma^2\_e}{2})} \\\ &= \text{exp}(\beta\_1)  
\end{align\*}  
$$

and similarly for the marginal outcomes

$$  
\begin{align\*}  
\frac{\text{E}(Y\_{ij} \mid \text{TX} = 1)}{\text{E}(Y\_{ij} \mid \text{TX} = 0)} &= \frac{\text{exp}(\beta\_0 + \beta\_1 + \frac{\sigma^2\_u + \sigma^2\_e}{2})}  
{\text{exp}(\beta\_0 + \frac{\sigma^2\_u + \sigma^2\_e}{2})} \\\ &= \text{exp}(\beta\_1).  
\end{align\*}  
$$

The cluster-specific and population-average effect is the same. However, this is not true in general for nonlinear transformation or GLMM link functions (which we shall see in part 2 on binary outcomes).

**95 % CIs for the backtransformed marginal effects**

Getting back to our fitted model, let’s see how we can transform the estimates into the posterior distribution of e.g. the marginal mean outcome in the control group. We do this by applying the back-transformation and marginalization using all the posterior samples of \(\hat\beta\_0\), \(\hat\sigma\_u\), and \(\hat\sigma\_e\).

marg\_f <- function(B0, u0, sd\_log) {

data.frame("marginal" = exp(B0 + (u0^2 + sd\_log^2)/2),

"conditional" = exp(B0 + (sd\_log^2)/2)

)

}

ss <- posterior\_samples(fit)

# mapply would also work

mu <- pmap\_df(

list(

B0 = ss$b\_Intercept,

u0 = ss$sd\_cluster\_\_Intercept,

sd\_log = ss$sigma

),

.f = marg\_f

)

map(mu, quantile, probs = c(0.025, 0.5, 0.975)) %>%

do.call(rbind, .)

## 2.5% 50% 97.5%

## marginal 454.4591 697.9116 1123.3794

## conditional 374.4601 570.5069 847.8781

**Summarizing the model**

Let’s summarize all the results from the model, both the conditional and marginal, absolute and multiplicative treatment effects.

ss <- posterior\_samples(fit, pars = c("b\_",

"sd",

"sigma"))

res <- ss %>%

select(B0 = b\_Intercept, B1 = b\_TX, u0 = sd\_cluster\_\_Intercept, sigma) %>%

mutate(y\_cond\_cc = exp(B0 + sigma^2/2),

y\_cond\_tx = exp(B0 + B1 + sigma^2/2),

y\_marg\_cc = exp(B0 + (u0^2 + sigma^2)/2),

y\_marg\_tx = exp(B0 + B1 + (u0^2 + sigma^2)/2),

TE\_cond\_ATE = y\_cond\_tx - y\_cond\_cc,

TE\_marg\_ATE = y\_marg\_tx - y\_marg\_cc,

TE\_cond\_multi = y\_cond\_tx/y\_cond\_cc,

TE\_marg\_multi = y\_marg\_tx/y\_marg\_cc,

TE\_multi = exp(B1)

) %>%

select(-B0, -B1, -u0, -sigma)

tab <- res %>%

gather(key, val) %>%

mutate(key = factor(key, levels = unique(key))) %>%

group\_by(key) %>%

do( data.frame(posterior\_median = median(.$val),

l\_95\_CI = quantile(.$val, 0.025),

u\_95\_CI = quantile(.$val, 0.975),

stringsAsFactors = FALSE)) %>%

ungroup() %>%

mutate(key = c("$\\text{E}(Y\_{ij} \\mid u\_i = 0, \\text{TX} = 0)$",

"$\\text{E}(Y\_{ij} \\mid u\_i = 0, \\text{TX} = 1)$",

"$\\text{E}(Y\_{ij} \\mid \\text{TX} = 0)$",

"$\\text{E}(Y\_{ij} \\mid \\text{TX} = 1)$",

"$\\text{E}(Y\_{ij} \\mid u\_i = 0, \\text{TX} = 1) - \\\\ \\text{E}(Y\_{ij} \\mid u\_i = 0, \\text{TX} = 0)$",

"$\\text{E}(Y\_{ij} \\mid \\text{TX} = 1) - \\\\ \\text{E}(Y\_{ij} \\mid \\text{TX} = 0)$",

"$\\frac{\\text{E}(Y\_{ij} \\mid u\_i = 0, \\text{TX} = 1)}{\\text{E}(Y\_{ij} \\mid u\_i = 0, \\text{TX} = 0)}$",

"$\\frac{\\text{E}(Y\_{ij} \\mid \\text{TX} = 1)}{\\text{E}(Y\_{ij} \\mid \\text{TX} = 0)}$",

"$\\text{exp}(\\beta\_1)$"),

Effect = c("Average outcome for a median cluster in the control group",

"Average outcome for a median cluster in the treatment group",

"Average outcome in the control group",

"Average outcome in the treatment group",

"Difference between the average outcome in a median cluster in the treatment versus control group",

"Difference between the average outcome in the treatment versus control group",

"Ratio of the average outcome in a median cluster in the treatment and the control group",

"Ratio of the average outcome in the treatment and the control group",

"Ratio of the average outcome in the treatment and the control group")) %>%

select(Effect, Formula = key, posterior\_median, l\_95\_CI, u\_95\_CI)

| **Effect** | **Formula** | **posterior\_median** | **l\_95\_CI** | **u\_95\_CI** |
| --- | --- | --- | --- | --- |
| Average outcome for a median cluster in the control group | \(\text{E}(Y\_{ij} \mid u\_i = 0, \text{TX} = 0)\) | 570.5 | 374.5 | 847.9 |
| Average outcome for a median cluster in the treatment group | \(\text{E}(Y\_{ij} \mid u\_i = 0, \text{TX} = 1)\) | 281.8 | 186.7 | 435.2 |
| Average outcome in the control group | \(\text{E}(Y\_{ij} \mid \text{TX} = 0)\) | 697.9 | 454.5 | 1123.4 |
| Average outcome in the treatment group | \(\text{E}(Y\_{ij} \mid \text{TX} = 1)\) | 342.7 | 231.3 | 575.5 |
| Difference between the average outcome in a median cluster in the treatment versus control group | \(\text{E}(Y\_{ij} \mid u\_i = 0, \text{TX} = 1) – \\ \text{E}(Y\_{ij} \mid u\_i = 0, \text{TX} = 0)\) | -286.1 | -580.2 | -37.9 |
| Difference between the average outcome in the treatment versus control group | \(\text{E}(Y\_{ij} \mid \text{TX} = 1) – \\ \text{E}(Y\_{ij} \mid \text{TX} = 0)\) | -349.0 | -747.5 | -47.4 |
| Ratio of the average outcome in a median cluster in the treatment and the control group | \(\frac{\text{E}(Y\_{ij} \mid u\_i = 0, \text{TX} = 1)}{\text{E}(Y\_{ij} \mid u\_i = 0, \text{TX} = 0)}\) | 0.5 | 0.3 | 0.9 |
| Ratio of the average outcome in the treatment and the control group | \(\frac{\text{E}(Y\_{ij} \mid \text{TX} = 1)}{\text{E}(Y\_{ij} \mid \text{TX} = 0)}\) | 0.5 | 0.3 | 0.9 |
| Ratio of the average outcome in the treatment and the control group | \(\text{exp}(\beta\_1)\) | 0.5 | 0.3 | 0.9 |

**Using brms::fitted**

Let’s compare our results to predicted means we get with brms::fitted and brms::marginal\_effects

## Estimate Est.Error Q2.5 Q97.5

## [1,] 570.5069 114.8815 374.4601 847.8781

## TX y cluster cond\_\_ estimate\_\_ se\_\_ lower\_\_ upper\_\_

## 1 0 481.4254 NA 1 570.5069 114.8815 374.4601 847.8781

We see that this is exactly the same as the median outcome (cluster-specific outcome) in the control group. I’ve seen some people confused about this since the function is called marginal\_effects.

**Limiting inference to only the included clusters**

We can also use fitted to average over the included clusters, this is very similar to fitting the clusters as “fixed effects”. So let’s compare averaging over the included clusters in the multilevel model to fitting a fixed effects (no pooling) model.

fit\_fixed <- brm(y ~ 0 + factor(cluster),

family = lognormal(),

data = d,

cores = 4,

iter = 2000)

Let’s look at the predicted mean for the control group using the fixed effects model.

cluster\_fixed <- fitted(fit\_fixed,

newdata = expand.grid(TX = 0,

cluster = 1:10),

re\_formula = NULL,

summary = FALSE)

cluster\_fixed <- apply(cluster\_fixed, 1, mean)

quantile(cluster\_fixed, c(0.025, 0.5, 0.975))

## 2.5% 50% 97.5%

## 600.4927 638.8134 682.9103

Which is really similar to the control groups sample mean which is 636. Now let’s compare the posterior distribution from the fixed effects model, to using our multilevel model to average over only the included clusters (and not the estimated distribution of clusters).

get\_fixed\_cluster\_marginal <- function(fit, newdat, label) {

cluster\_fixed\_ <- fitted(fit,

newdata = newdat,

re\_formula = NULL,

summary = FALSE)

cluster\_fixed <- apply(cluster\_fixed\_, 1, mean)

# posterior samples

list(ss\_fixed = cluster\_fixed\_,

ss\_marg = cluster\_fixed,

label = label)

}

# Get prediction for each treatment group

fixed\_cluster\_C\_RE <- get\_fixed\_cluster\_marginal(fit = fit,

newdat = expand.grid(TX = 0,

cluster = 1:10),

label = "C\_RE")

fixed\_cluster\_TX\_RE <- get\_fixed\_cluster\_marginal(fit = fit,

newdat = expand.grid(TX = 1,

cluster = 11:20),

label = "TX\_RE")

fixed\_cluster\_C\_FE <- get\_fixed\_cluster\_marginal(fit = fit\_fixed,

newdat = expand.grid(TX = 0,

cluster = 1:10),

label = "C\_FE")

fixed\_cluster\_TX\_FE <- get\_fixed\_cluster\_marginal(fit = fit\_fixed,

newdat = expand.grid(TX = 1,

cluster = 11:20),

label = "TX\_FE")

# Summarise posterior median and 95 % CI

sums <- data.frame("model" = rep(c("RE",

"FE"), 3),

"variable" = rep(c("Control",

"Treatment",

"Difference"), each = 2))

qs <- rbind(quantile(fixed\_cluster\_C\_RE$ss\_marg, c(0.025, 0.5, 0.975)),

quantile(fixed\_cluster\_C\_FE$ss\_marg, c(0.025, 0.5, 0.975)),

quantile(fixed\_cluster\_TX\_RE$ss\_marg, c(0.025, 0.5, 0.975)),

quantile(fixed\_cluster\_TX\_FE$ss\_marg, c(0.025, 0.5, 0.975)),

quantile(fixed\_cluster\_TX\_RE$ss\_marg - fixed\_cluster\_C\_RE$ss\_marg, c(0.025, 0.5, 0.975)),

quantile(fixed\_cluster\_TX\_FE$ss\_marg - fixed\_cluster\_C\_FE$ss\_marg, c(0.025, 0.5, 0.975)))

colnames(qs) <- c("l\_95\_CI", "median", "u\_95\_CI")

sums <- cbind(sums, qs)

ps <- rbind(data.frame("model" = "RE", variable = "Control", samples = fixed\_cluster\_C\_RE$ss\_marg),

data.frame("model" = "RE", variable = "Treatment", samples = fixed\_cluster\_TX\_RE$ss\_marg),

data.frame("model" = "RE", variable = "Difference", samples = fixed\_cluster\_TX\_RE$ss\_marg - fixed\_cluster\_C\_RE$ss\_marg),

data.frame("model" = "FE", variable = "Control", samples = fixed\_cluster\_C\_FE$ss\_marg),

data.frame("model" = "FE", variable = "Treatment", samples = fixed\_cluster\_TX\_FE$ss\_marg),

data.frame("model" = "FE", variable = "Difference", samples = fixed\_cluster\_TX\_FE$ss\_marg - fixed\_cluster\_C\_FE$ss\_marg))

# plot

p <- ggplot(ps, aes(x = samples,

color = model,

fill = model)) +

geom\_density(alpha = 0.25, aes(y = ..scaled..)) +

geom\_segment(data = sums,

aes(x = l\_95\_CI,

xend = u\_95\_CI,

y = 0,

yend = 0),

position = position\_nudge(y = c(-0.025, -0.05))) +

geom\_point(data = sums,

aes(x = median,

y = 0),

position = position\_nudge(y = c(-0.025, -0.05))) +

facet\_wrap(~variable, scales = "free\_x") +

labs(x = "Posterior samples",

y = "Cluster mean ($)",

title = "Treating clusters as \"fixed\"",

subtitle = "Comparing no pooling (FE) vs partial pooling (RE)\nwhen both models limit inferences to the included clusters only"

) +

theme\_minimal() +

theme(legend.position = "bottom")

We see that the results are highly similar. Although, the models behind the results do differ. If we look at the estimates for the specific clusters we can see the—in this model very weak—effect of partial pooling in the multilevel model.

**Intraclass correlations (ICC)**

Since the variance components are on the log scale ICCs calculated using them will also be on the log scale. This is not always what we want, for instance, if we are calculating agreement, reliability, or heritability, it can make a lot of sense to calculate ICCs on the data scale. If I am calculating ICCs for family members’ reports on gambling losses, then I am interested in agreement on the data scale (USD) and not on the log scale.

ICCs on the data scale can be calculated using the same techniques we used for the marginalization of the outcomes. You can read more about this in the context of GLMMs in Villemereuil et al. (2016; and the related R package QCglmm). Here is an example, first we calculate the marginal mean

$$ \text{E}(y\_{i} \mid TX\_i) = \bar\mu\_i = \int \text{exp}(x + \sigma\_e^2/2) f\_{\mathcal{N}}(x, \mu\_{i}, \sigma\_{u\_0}^2)dx$$

then we use the marginal mean to calculate the expected variance between clusters on the data scale,

$$ \sigma\_B^2 = \int (e^x – \bar\mu)^2 f\_{\mathcal{N}}(x, \mu\_{i}, \sigma\_{u\_0^2})dx$$

and lastly we calculate the expected variance within clusters on the data scale,

$$ \sigma\_W^2 = \int \left(( \text{exp}(\sigma\_e^2) – 1 ) \text{exp}(2x + \sigma\_e^2) \right)f\_{\mathcal{N}}(x, \mu\_i, \sigma\_{u\_0^2})dx.$$

Where \((\text{exp}(\sigma\_e^2) – 1 ) \text{exp}(2x + \sigma\_e^2)\) is the variance of the lognormal distribution evaluated at \(x\).

ICC on response scale in then

$$ \text{Cor}(y\_{ij}, y\_{ij’}) = \frac{\sigma\_B^2}{\sigma\_B^2 + \sigma\_W^2} $$

Whereas, the log scale ICC is,

$$ \text{Cor}\left(\text{log}(y\_{ij}), \text{log}(y\_{ij’})\right) = \frac{\sigma\_u^2}{\sigma\_u^2 + \sigma\_e^2} $$

Since this is also a one-dimensional integral we can use integrate,

get\_ICC\_ln <- function(B0, u0, sd\_log, ...) {

mu <- integrate(

f = function(x) {

exp(x + sd\_log^2 / 2) \* dnorm(x, B0, sd = u0)

},

lower = B0 - 10 \* u0,

upper = B0 + 10 \* u0

)$value

sigma2\_bw <- integrate(

f = function(x) {

(exp(x + sd\_log^2 / 2) - mu)^2 \* dnorm(x, B0, sd = u0)

},

lower = B0 - 10 \* u0,

upper = B0 + 10 \* u0

)$value

sigma2\_wi <- integrate(

f = function(x) {

(exp(sd\_log^2) - 1) \* exp(2 \* x + sd\_log^2) \* dnorm(x, B0, sd = u0)

},

lower = B0 - 10 \* u0,

upper = B0 + 10 \* u0

)$value

ICC <- sigma2\_bw / (sigma2\_bw + sigma2\_wi)

list(

"ICC" = ICC,

"WI\_sd" = sqrt(sigma2\_wi),

"BW\_sd" = sqrt(sigma2\_bw),

"mean" = mu

)

}

If you find code easier to read, the Monte Carlo version would be

get\_ICC\_ln\_sim <- function(B0, u0, sd\_log, R = 1e4) {

RE <- B0 + rnorm(R, 0, u0)

y <- exp(RE + sd\_log^2/2)

sigma2\_bw <- var(y)

sigma2\_wi <- mean( (exp(sd\_log^2) - 1) \* exp(2 \* RE + sd\_log^2))

ICC <- sigma2\_bw / (sigma2\_bw + sigma2\_wi)

list("ICC" = ICC,

"WI\_sd" = sqrt(sigma2\_wi),

"BW\_sd" = sqrt(sigma2\_bw),

"mean" = mean(y))

}

Let’s see if the two methods agree.

ICCS <- list(get\_ICC\_ln(B0,

u0,

sd\_log),

get\_ICC\_ln\_sim(B0,

u0,

sd\_log,

R = 1e6),

data.frame(ICC = u0^2/(u0^2 + sd\_log^2),

WI\_sd = NA,

BW\_sd = NA,

mean = NA)

)

res <- lapply(ICCS, data.frame) %>%

do.call(rbind, .)

|  | **Data scale (Integrate)** | **Data scale (Simulate)** | **Log scale** |
| --- | --- | --- | --- |
| ICC | 0.44 | 0.44 | 0.5 |
| WI\_sd | 387.71 | 387.69 | NA |
| BW\_sd | 342.15 | 342.22 | NA |
| mean | 642.01 | 641.93 | NA |

Great! We also see that by calculating variances on the data scale the ICCs are lower, in this case, the difference is small, but if the log variance is large this difference can be *very* large. So when you see ICCs from lognormal models or GLMMs you should ask if they are on the transformed (latent) scale or on the observed (data) scale, and if the scale is approriate for the research question.

**Simulation study**

I began by promising that the methods in this post could make your *frequentist* life easier, and then continued to only fit Bayesian models. So let’s do a small simulation study to evaluate the frequentist properties of this model and all the transformations and marginalizations. I will try two different models, shown in the table below, each model will be evaluated using 5000 replications, and I will fit both the correct model using a lognormal response distribution and a standard LMM using the Gaussian response distribution.

| **Parameter** | **Sim1: less skew** | **Sim2: more skew** |
| --- | --- | --- |
| \(\beta\_0\) | log(500) | log(500) |
| \(\beta\_1\) | log(0.5) | log(0.5) |
| \(\sigma\_u\) | 0.5 | 1 |
| \(\sigma\_e\) | 0.5 | 1 |
| \(n\_1\) | 30 | 5 |
| \(n\_2\) | 10 | 75 |

**Simulation results**

In the figures below we see that the posterior median show mostly negligible bias for both the original parameters and the transformed marginal effects. The Gaussian model has the most trouble with the variance components, which show substantial bias.

If we look at the coverage probabilities we see that the lognormal model’s posterior intervals have nominal coverage rates for all parameters. The Gaussian model has slightly worse coverage for the treatment effects, but not that bad. However, the CI’s from the Gaussian model does a poor job of capturing the true ICC and variance components.

Normal model

| **var** | **model** | **est\_mean** | **est\_median** | **theta** | **est\_rel\_bias** | **est\_median\_rel\_bias** | **se** | **coverage** | **power** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ICC | gaussian | 0.4 | 0.4 | 0.4 | 0.0 | 0.0 | 0.1 | 0.9 | 1.0 |
| marg\_TE | gaussian | -323.5 | -323.4 | NA | NA | NA | 125.0 | NA | 0.8 |
| marg\_mean\_C | gaussian | 635.6 | 635.9 | 642.0 | 0.0 | 0.0 | 105.2 | 0.9 | 1.0 |
| marg\_mean\_TX | gaussian | 312.1 | 312.4 | 321.0 | 0.0 | 0.0 | 53.6 | 1.0 | 1.0 |
| multiplicative\_TE\_ratio\_marg | gaussian | 0.5 | 0.5 | 0.5 | 0.0 | 0.0 | 0.1 | 1.0 | 1.0 |
| sd\_BW\_C | gaussian | 267.3 | 261.3 | 342.2 | -0.2 | -0.2 | 84.8 | 0.5 | 1.0 |
| sd\_BW\_TX | gaussian | 267.3 | 261.3 | 171.1 | 0.6 | 0.5 | 84.8 | 0.4 | 1.0 |
| sd\_WI\_C | gaussian | 302.7 | 302.4 | 387.7 | -0.2 | -0.2 | 57.0 | 0.1 | 1.0 |
| sd\_WI\_TX | gaussian | 302.7 | 302.4 | 193.9 | 0.6 | 0.6 | 57.0 | 0.0 | 1.0 |

Lognormal model

| **var** | **model** | **est\_mean** | **est\_median** | **theta** | **est\_rel\_bias** | **est\_median\_rel\_bias** | **se** | **coverage** | **power** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| b0 | LN | 6.2 | 6.2 | 6.2 | 0.0 | 0.0 | 0.2 | 1.0 | 1.0 |
| b1 | LN | -0.7 | -0.7 | -0.7 | 0.0 | 0.0 | 0.2 | 0.9 | 0.8 |
| ICC | LN | 0.4 | 0.4 | 0.4 | 0.0 | 0.0 | 0.1 | 0.9 | 1.0 |
| ICC\_log | LN | 0.5 | 0.5 | 0.5 | 0.0 | 0.0 | 0.1 | 0.9 | 1.0 |
| marg\_TE | LN | -342.1 | -330.4 | NA | NA | NA | 126.7 | NA | 0.8 |
| marg\_mean\_C | LN | 683.1 | 664.0 | 642.0 | 0.1 | 0.0 | 118.8 | 0.9 | 1.0 |
| marg\_mean\_TX | LN | 341.0 | 331.5 | 321.0 | 0.1 | 0.0 | 58.0 | 1.0 | 1.0 |
| median\_clust\_TE | LN | -258.5 | -253.1 | NA | NA | NA | 94.2 | NA | 0.8 |
| median\_clust\_C | LN | 516.2 | 508.3 | 500.0 | 0.0 | 0.0 | 84.4 | 1.0 | 1.0 |
| median\_clust\_TX | LN | 257.7 | 253.7 | 250.0 | 0.0 | 0.0 | 41.3 | 1.0 | 1.0 |
| multiplicative\_TE\_exp\_beta1 | LN | 0.5 | 0.5 | 0.5 | 0.1 | 0.0 | 0.1 | 0.9 | 1.0 |
| multiplicative\_TE\_ratio\_marg | LN | 0.5 | 0.5 | 0.5 | 0.1 | 0.0 | 0.1 | 0.9 | 1.0 |
| multiplicative\_TE\_ratio\_medians | LN | 0.5 | 0.5 | 0.5 | 0.1 | 0.0 | 0.1 | 0.9 | 1.0 |
| sd\_BW\_C | LN | 409.0 | 370.9 | 342.2 | 0.2 | 0.1 | 131.4 | 0.9 | 1.0 |
| sd\_BW\_TX | LN | 204.1 | 185.1 | 171.1 | 0.2 | 0.1 | 65.0 | 0.9 | 1.0 |
| sd\_log | LN | 0.5 | 0.5 | 0.5 | 0.0 | 0.0 | 0.0 | 0.9 | 1.0 |
| sd\_u0\_log | LN | 0.5 | 0.5 | 0.5 | 0.1 | 0.0 | 0.1 | 0.9 | 1.0 |
| sd\_WI\_C | LN | 429.6 | 409.3 | 387.7 | 0.1 | 0.1 | 88.8 | 0.9 | 1.0 |
| sd\_WI\_TX | LN | 214.4 | 204.3 | 193.9 | 0.1 | 0.1 | 43.5 | 0.9 | 1.0 |

**Power**

Lastly, let’s take a closer look at the marginalized treatment effect on the data scale. The figure below shows a sample of the confidence intervals for the difference in the average money lost gambling from the two conditions, from Model 2 with more skewed data. We can see that the, although, the normal model is quite robust and the estimated effect is unbiased, it is less precise which leads to reduced power and CIs with worse coverage compared to the true lognormal model.

**Summary**

There is a lot of information in this post, so let’s summarize the key take-home messages:

* Both cluster-specific (conditional) and population-average (marginal) treatment effects are useful—they just answer different questions and we should be careful to not mixed them up.
* The multilevel model can be used to give both cluster-specific (conditional) and population-average effects on both the original and transformed scale while retaining all of the advantages of likelihood-based/Bayesian methods.
* There is no free lunch. We saw that the calculations of the marginal effects on the data scale use estimates of both the lognormal variance, as well as the random intercept Thus, model misspecification will have a large impact on the appropriateness of the back-transformation. In that sense, directly modeling the population-average effects using e.g. GEE and applying nonparametric retransformations (e.g. smearing) is more robust in some aspects. However, once we add complexities like MNAR missing data in combination with weak adherence (both by patients and therapists…) to complex interventions, then it can be argued that a carefully specified generative model that combines both statistical and clinical expertise will be more likely to give useful answers regarding the treatment effect. In my experience, such models will often have multilevel and cluster-specific components.
* For outcomes that are highly meaningful on the original scale (such as expenditure), absolute effects on the data scale can be very meaningful.
* Using the default priors in brms resulted in frequentist CI’s with nominal coverage probabilities.
* It is useful to understand how the conditional and marginal effects relate to each other, to avoid misinterpretations of the default cluster-specific effects (very common in the clinical studies I read).

**Further reading**

The issues covered in this post is dealt with in many articles, here is a selection of relevant articles:

* Gromping, U. (1996). A Note on Fitting a Marginal Model to Mixed Effects Log-Linear Regression Data via GEE. Biometrics, 52(1), 280–285. https://doi.org/10.2307/2533162
* Heagerty, P. J., & Zeger, S. L. (2000). Marginalized multilevel models and likelihood inference (with comments and a rejoinder by the authors). Statistical Science, 15(1), 1–26. https://doi.org/10.1214/ss/1009212671
* Hedeker, D., Toit, S. H. C. du, Demirtas, H., & Gibbons, R. D. (2018). A note on marginalization of regression parameters from mixed models of binary outcomes. Biometrics, 74(1), 354–361. https://doi.org/10.1111/biom.12707
* Villemereuil, P. de, Schielzeth, H., Nakagawa, S., & Morrissey, M. (2016). General Methods for Evolutionary Quantitative Genetic Inference from Generalized Mixed Models. Genetics, 204(3), 1281–1294. https://doi.org/10.1534/genetics.115.186536

**Simulation code**

run\_sim.R: perform the simulation

library(tidyr)

library(dplyr)

library(brms)

library(future.apply)

library(purrr)

source("sim/functions.R")

# Parallelization setup

cores <- 16

nsim <- 5000

plan(tweak(multiprocess,

workers = cores))

# ---- model-setup ----

set.seed(4445)

pars <- list("n1" = 30, # observations per cluster

"n2" = 10, # clusters per treatment

"B0" = log(500),

"B1" = log(0.5),

"sd\_log" = 0.5,

"u0" = 0.5)

d <- do.call(gen\_data,

pars)

# ---- compile models ----

fit\_LN <- brm(y ~ 1 + TX + (1 | cluster),

family = lognormal(),

data = d,

cores = 1,

iter = 1)

fit\_gauss <- brm(y ~ 1 + TX + (1 | cluster),

family = gaussian(),

data = d,

cores = 1,

iter = 1)

# Run ---------------------------------------------------------------------

res1 <- run\_sim(nsim = nsim,

pars = pars,

fit\_LN = fit\_LN,

fit\_gauss = fit\_gauss)

pars <- list("n1" = 5,

"n2" = 75,

"B0" = log(500),

"B1" = log(0.5),

"sd\_log" = 1,

"u0" = 1)

res2 <- run\_sim(nsim = nsim,

pars = pars,

fit\_LN = fit\_LN,

fit\_gauss = fit\_gauss)

saveRDS(res1, file = "sim\_res\_GLMM\_LN1.rds")

saveRDS(res2, file = "sim\_res\_GLMM\_LN2.rds")

summarise\_sim.R: Summarize results

library(tidyr)

library(dplyr)

library(ggplot2)

source("sim/functions.R")

res1 <- readRDS("sim\_res\_GLMM\_LN1.rds")

res2 <- readRDS("sim\_res\_GLMM\_LN2.rds")

# summary

res\_sum1 <- summarise\_sim(res1)

res\_sum2 <- summarise\_sim(res2)

functions.R: Functions used

# ---- gen-data ----

#' Generate lognormal data with a random intercept

#'

#' @param n1 patients per cluster

#' @param n2 clusters per treatment

#' @param B0 log intercept

#' @param B1 log treatment effect

#' @param sd\_log log sd

#' @param u0 SD of log intercepts (random intercept)

#'

#' @return a data.frame

gen\_data <- function(n1, n2, B0, B1, sd\_log, u0) {

cluster <- rep(1:(2 \* n2), each = n1)

TX <- rep(c(0, 1), each = n1 \* n2)

u0 <- rnorm(2 \* n2, sd = u0)[cluster]

mulog <- (B0 + B1 \* TX + u0)

y <- rlnorm(2 \* n1 \* n2, meanlog = mulog, sdlog = sd\_log)

d <- data.frame(cluster,

TX,

y)

d

}

# ---- icc-integrate ----

get\_ICC\_ln <- function(B0, u0, sd\_log, ...) {

mu <- integrate(

f = function(x) {

exp(x + sd\_log^2 / 2) \* dnorm(x, B0, sd = u0)

},

lower = B0 - 10 \* u0,

upper = B0 + 10 \* u0

)$value

sigma2\_bw <- integrate(

f = function(x) {

(exp(x + sd\_log^2 / 2) - mu)^2 \* dnorm(x, B0, sd = u0)

},

lower = B0 - 10 \* u0,

upper = B0 + 10 \* u0

)$value

sigma2\_wi <- integrate(

f = function(x) {

(exp(sd\_log^2) - 1) \* exp(2 \* x + sd\_log^2) \* dnorm(x, B0, sd = u0)

},

lower = B0 - 10 \* u0,

upper = B0 + 10 \* u0

)$value

ICC <- sigma2\_bw / (sigma2\_bw + sigma2\_wi)

list(

"ICC" = ICC,

"WI\_sd" = sqrt(sigma2\_wi),

"BW\_sd" = sqrt(sigma2\_bw),

"mean" = mu

)

}

# ---- icc-sim ----

get\_ICC\_ln\_sim <- function(B0, u0, sd\_log, R = 1e4) {

RE <- B0 + rnorm(R, 0, u0)

y <- exp(RE + sd\_log^2/2)

sigma2\_bw <- var(y)

sigma2\_wi <- mean( (exp(sd\_log^2) - 1) \* exp(2 \* RE + sd\_log^2))

ICC <- sigma2\_bw / (sigma2\_bw + sigma2\_wi)

list("ICC" = ICC,

"WI\_sd" = sqrt(sigma2\_wi),

"BW\_sd" = sqrt(sigma2\_bw),

"mean" = mean(y))

}

# ---- sim-function ----

summarise\_posterior <- function(samples, name) {

data.frame(

"var" = name,

"est" = mean(samples),

"est\_median" = median(samples),

"est\_se" = sd(samples),

"lwr" = quantile(samples, 0.025),

"upr" = quantile(samples, 0.975),

row.names = NULL,

stringsAsFactors = FALSE

)

}

add\_thetas <- function(x, pars) {

theta\_C <- do.call(get\_ICC\_ln, pars)

theta\_TX <- get\_ICC\_ln(B0 = pars$B0 + pars$B1,

u0 = pars$u0,

sd\_log = pars$sd\_log)

mutate(x,

theta = case\_when(

var == "b0" ~ pars$B0,

var == "b1" ~ pars$B1,

var == "sd\_u0\_log" ~ pars$u0,

var == "sd\_BW\_C" ~ theta\_C$BW\_sd,

var == "sd\_BW\_TX" ~ theta\_TX$BW\_sd,

var == "sd\_WI\_C" ~ theta\_C$WI\_sd,

var == "sd\_WI\_TX" ~ theta\_TX$WI\_sd,

var == "marg\_mean\_C" ~ theta\_C$mean,

var == "marg\_mean\_TX" ~ theta\_TX$mean,

var == "marg\_TE" ~ theta\_TX$mean - theta\_C$mean,

var == "multiplicative\_TE\_ratio\_marg" ~ theta\_TX$mean/theta\_C$mean,

var == "multiplicative\_TE\_ratio\_medians" ~ with(pars, exp(B0 + B1)/exp(B0)),

var == "multiplicative\_TE\_exp\_beta1" ~ exp(pars$B1),

var == "median\_clust\_C" ~ with(pars, exp(B0)),

var == "median\_clust\_TX" ~ with(pars, exp(B0 + B1)),

var == "median\_clust\_TE" ~ with(pars, exp(B0 + B1) - exp(B0)),

var == "sd\_log" ~ pars$sd\_log,

var == "ICC" ~ theta\_C$ICC,

var == "ICC\_log" ~ with(pars, u0^2/(u0^2 + sd\_log^2)),

TRUE ~ NA\_real\_

)

)

}

# extract results

extract\_results\_LN <- function(fit\_LN) {

ss <- posterior\_samples(fit\_LN)

icc\_C <- pmap\_df(

list(

B0 = ss$b\_Intercept,

u0 = ss$sd\_cluster\_\_Intercept,

sd\_log = ss$sigma

),

.f = get\_ICC\_ln

)

icc\_TX <- pmap\_df(

list(

B0 = ss$b\_Intercept + ss$b\_TX,

u0 = ss$sd\_cluster\_\_Intercept,

sd\_log = ss$sigma

),

.f = get\_ICC\_ln

)

# posterior quantities

median\_clust\_C <- exp(ss$b\_Intercept)

median\_clust\_TX <- exp(ss$b\_Intercept + ss$b\_TX)

vars <- list(

"b0" = ss$b\_Intercept,

"b1" = ss$b\_TX,

"sd\_u0\_log" = ss$sd\_cluster\_\_Intercept,

"sd\_BW\_C" = icc\_C$BW\_sd,

"sd\_BW\_TX" = icc\_TX$BW\_sd,

"sd\_WI\_C" = icc\_C$WI\_sd,

"sd\_WI\_TX" = icc\_TX$WI\_sd,

"marg\_mean\_C" = icc\_C$mean,

"marg\_mean\_TX" = icc\_TX$mean,

"marg\_TE" = icc\_TX$mean - icc\_C$mean,

"median\_clust\_C" = median\_clust\_C,

"median\_clust\_TX" = median\_clust\_TX,

"median\_clust\_TE" = median\_clust\_TX - median\_clust\_C,

"multiplicative\_TE\_ratio\_marg" = icc\_TX$mean/icc\_C$mean,

"multiplicative\_TE\_ratio\_medians" = median\_clust\_TX/median\_clust\_C,

"multiplicative\_TE\_exp\_beta1" = exp(ss$b\_TX),

"sd\_log" = ss$sigma,

"ICC" = icc\_C$ICC,

"ICC\_log" = ss$sd\_cluster\_\_Intercept^2/(ss$sd\_cluster\_\_Intercept^2 + ss$sigma^2)

)

res\_LN <- map2\_df(vars,

names(vars),

summarise\_posterior)

res\_LN$model <- "LN"

res\_LN

}

# gaussian

extract\_results\_gauss <- function(fit\_gauss) {

ss <- posterior\_samples(fit\_gauss)

# posterior quantities

vars <- list(

"sd\_BW\_C" = ss$sd\_cluster\_\_Intercept,

"sd\_BW\_TX" = ss$sd\_cluster\_\_Intercept,

"sd\_WI\_C" = ss$sigma,

"sd\_WI\_TX" = ss$sigma,

"marg\_mean\_C" = ss$b\_Intercept,

"marg\_mean\_TX" = ss$b\_Intercept + ss$b\_TX,

"marg\_TE" = ss$b\_TX,

"multiplicative\_TE\_ratio\_marg" = (ss$b\_Intercept + ss$b\_TX)/ss$b\_Intercept,

"ICC" = ss$sd\_cluster\_\_Intercept^2/(ss$sd\_cluster\_\_Intercept^2 + ss$sigma^2)

)

res\_gauss <- map2\_df(vars,

names(vars),

summarise\_posterior)

res\_gauss$model <- "gaussian"

res\_gauss

}

# performs 1 sim replications

refit\_mod <- function(i, fit\_LN, fit\_gauss, pars) {

d <- do.call(gen\_data, pars)

# fit models

fit\_LN <- update(

fit\_LN,

newdata = d,

recompile = FALSE,

chains = 1,

iter = 2000

)

fit\_gauss <- update(

fit\_gauss,

newdata = d,

recompile = FALSE,

chains = 1,

iter = 2000

)

# save results

res\_LN <- extract\_results\_LN(fit\_LN)

res\_gauss <- extract\_results\_gauss(fit\_gauss)

# Combine

res <- rbind(res\_LN,

res\_gauss)

res$sim <- i

res

}

# Run simulation ----------------------------------------------------------

run\_sim <- function(nsim, pars, fit\_LN, fit\_gauss) {

res <- future.apply::future\_lapply(1:nsim,

refit\_mod,

fit = fit\_LN,

fit\_gauss = fit\_gauss,

pars = pars)

res <- do.call(rbind, res)

list(res = res,

pars = pars)

}

# Summarise simulation ----------------------------------------------------

summarise\_sim <- function(x) {

x$res %>%

add\_thetas(x$pars) %>%

group\_by(var, model) %>%

summarise(

est\_mean = mean(est),

est\_median = mean(est\_median),

theta = unique(theta),

est\_rel\_bias = (est\_mean - theta) / theta,

est\_median\_rel\_bias = (est\_median - theta) / theta,

se = sd(est),

coverage = mean(lwr < theta & theta < upr),

power = mean(sign(lwr) == sign(upr))

)

}