Unbiased predictions of fixed variables in Generalized Mixed Models using **fixPredict** package

Maciej J. Dańko*[1]
[1]*Max Planck Institute for Demographic Research*

Abstract

to be added

1 Introduction

1.1 Generalized linear mixed effect models

The linear mixed models are defined as

$$y = X\beta + Zu + \epsilon \tag{1}$$

where, y is a vector of observations (dependent variable), β and u are vectors of unknown the fixed effect and the random effect coefficients, X and Z are design matrices relating the observed y to the the β and the u, and ϵ is an unknown vector of the random terms. It is assumed that expected values of u and ϵ are equal to zero ($E(u) = E(\epsilon) = 0$), hence the expectation of the response $E(y)$ is given by:

$$E(y) = X\beta + Zu \tag{2}$$

In the generalized linear mixed models, the response variable y may have different distribution than normal. The models use a link function g to transform the expectation of the response $E(y)$ to the linear predictor η:

$$g(E(y)) = \eta = X\beta + Zu \tag{3}$$

Similarly as in eq. 1, η is a linear function of regressors and random effects. Any link function must be invertable, so that $E(y)$ can be calculated.

$$E(y) = g^{-1}(X\beta + Zu) \tag{4}$$

1.2 Predictions in generalized linear mixed models

In the generalized linear models making the predictions is straightforward. Once the vector of coefficients $\hat{\beta}$ is estimated it can be use together with a new design matrix X_p to predict the marginal responses at particular value of covariates:

$$E(y_p) = g^{-1}(X_p\hat{\beta}) \tag{5}$$

In the generalized linear mixed models the predictions are more problematic because of the presence of Z_p.

*Corresponding author: danko@demogr.mpg.de, maciej.danko@gmail.com

$$E(y_p) = g^{-1}(X_p\hat{\beta} + Z_p\hat{u}) \tag{6}$$

The typical approach used across different packages is either to (1) ignore $Z_p\hat{u}$ (set it to zero) or (2) construct matrix Z_p for a particular grouping level of random variables such that matrices X_p and Z_p have the same number of rows.
Let us consider an exemplary data for a clinical trial of two competing oral antifungal treatments for the toenail infection (`HSAUR2` package), with `patientID` as random effect. The data can be fitted using `glmer` function of the `lme4` package

```
# install packages
if(!require(lme4)) {install.packages('lme4'); library(lme4)}
if(!any(installed.packages()[,1]=='HSAUR2')) install.packages('HSAUR2')

# load the clinical trial data for the toenail infection
data1 <- HSAUR2::toenail

# fit the generalized linear mixed model to binary outcome data
model1 <- glmer(outcome ~ treatment * visit+(1|patientID),
             data=data1, family=binomial, nAGQ=25)
```

As stated above, one of the way of model predictions is to exclude random terms. This can be done by calling a generic `predict` function with parameter `re.form = NA` for a model fitted with the `glmer` function (`lme4` package), or with parameter `level = 0` for model fitted with the `lme` function (`nlme` package). The syntax may differ for other packages.

```
newdata <- expand.grid(visit=c(1,4,7), treatment=levels(data1$treatment))
data.frame(newdata, predict = predict(model1, newdata=newdata,
                                      type='response', re.form=NA))
```

```
  visit    treatment      predict
1     1 itraconazole 0.2240253938
2     4 itraconazole 0.0261602678
3     7 itraconazole 0.0024933054
4     1  terbinafine 0.2107740970
5     4  terbinafine 0.0120927345
6     7  terbinafine 0.0005607344
```

Here I consider 1^{st}, 4^{th}, and 7^{th} visit in combination of two treatments. The predictions are performed on the scale of response variable (`type = 'response'`).
The alternative approach is to include the random effects for particular subjects or group of subjects. In `glmer` it is done by setting `re.form = NULL` or by specifying the random effect formula explicitly.

```
newdata <- data.frame(newdata, patientID=1:2)
data.frame(newdata, predict = predict(model1, newdata=newdata,
                                      type='response', re.form=~(1|patientID)))
```

```
  visit    treatment patientID     predict
1     1 itraconazole         1 0.915633988
2     4 itraconazole         2 0.182607465
3     7 itraconazole         1 0.085893643
4     1  terbinafine         2 0.689537664
5     4  terbinafine         1 0.315145593
6     7  terbinafine         2 0.004644215
```

1.3 Averaging fixed effects conditionally to the estimated population random effects

Here I present the alternative approach, which calculates fixed effects conditionally to the estimated population random effects $\hat{R} = Z\hat{u}$. In this approach I define $\hat{R}_c$ as a vector of all elements of $\hat{R}$ that belongs to a specific combination c of categorical variable levels. Each c correspond to a single row of X_p design matrix. The mechanism of grouping $\hat{R}$ into $\hat{R}_c$ is demonstrated in **Tab. 1**.

Table 1. Demonstration of grouping $\hat{R}$ into $\hat{R}_c$ for a data consistent of two 3-level categorical variables (`X1` and `X2`). The 28 estimated random effects are grouped according to combination of variable levels c. n_c is a number of individual random effects in a category c and P_c is a specific element of the vector $\hat{P} = X_p\hat{\beta}$

X1 level	X2 level	c	$\hat{R}_c$	n_c	$\hat{P}_c$
a	A	aA	{0.4567, 0.1456, -0.4537, -1.4561}	4	2.4567
b	A	bA	{0.1572, 1.2645}	2	3.8761
c	A	cA	{0.4464, 0.1455, -0.4537, -1.4561}	4	5.1411
a	B	aB	{0.4567, 0.1456, -1.4561}	3	1.0808
b	B	bB	{0.1572, 1.2645, -0.4537, 0.4567}	4	3.6432
c	B	cB	{0.1572, 1.2645}	2	2.2221
a	C	aC	{0.4464}	1	3.2512
b	C	bC	{0.4567, 0.1456, -0.4537, -1.4561}	4	3.1235
c	C	cC	{0.4567, 1.2645, -0.4537, -1.4561}	4	3.5378

The next step is to calculate averaged fixed effects for each c:

$$E(y_p(c)) = \sum_{i}^{n_c} g^{-1}\left(\hat{P}_c + \hat{R}_c(i)\right)/n_c \tag{7}$$

1.4 Averaging fixed effects conditionally to the missing variables

The `predict` function requires user to specify all variables used for model fitting in the `newdata` data frame. Yet, I propose an alternative approach, which use the idea of averaging presented in previous section. The effect of missing variables is coded as $S = X\hat{\beta_m}$, where X is design matrix for a total population, and β_m is a vector of model coefficients, which has inserted 0 for every coefficient related to variables included in the `newdata`.

Table 2. Demonstration of calculation of the $\hat{\beta_m}$ and its complement $\hat{\beta}_d$ for a model `Y ~ X1 + X2 + C1 + X1:X2`, where `X1` and `X2` are two-level categorical variables, `C1` is a continuous variable, and '`:`' denotes interaction. It is assumed that only `X1` and `C1` variables are present in the `newdata` data frame.

	(Intercept)	X1b	X2B	C1	X1b:X2B
$\hat{\beta}$	4.0122	0.1234	2.0567	0.1645	-0.8876
$\hat{\beta_m}$	0	0	2.0567	0	-0.8876
$\hat{\beta}_d$	4.0122	0.1234	0	0.1645	0

The complement $\hat{\beta}_d$ is then used to calculate predictions for variables present in the `newdata`: $\hat{Q} = X_p\hat{\beta}_d$. Finally both predictions can be combined in similar way as it was done for random effects (see eq. 7 and **Tab. 1**):

$$E(y_p(c)) = \sum_{i}^{n_c} g^{-1}\Big(\hat{Q}_c + \hat{S}_c(i)\Big)/n_c \tag{8}$$

1.5 Offset and rates

Offset ξ is a explenatory variable of a known effect. The offset is not estimated, but its values are simply added to the linear predictor.

$$g(E(y)) = \eta = X\beta + \xi \tag{9}$$

The offset is especially useful in GLMM when dependent variable is a count. In such a case offset variable is used to handle the rates and an offset ξ_i of observation i is related to its exposure e_i via link function $\xi_i = g(e_i)$, where g is typically a *log* function.
When the offset is used during estimation of a model, it should be also included in the model predictions. To include the offset for each combination of factors and their levels c, the classification method described in previous sections can be adopted.

$$E(y_p(c)) = \sum_{i}^{n_c} g^{-1}\Big(\hat{P}_c + \xi_c(i)\Big)/n_c \tag{10}$$

To calculate rates, sum of counts are dived by sum of exposures for each c.

$$E(y_p(c)) = \sum_{i}^{n_c} g^{-1}\Big(\hat{P}_c + \xi_c(i)\Big) / \sum_{i}^{n_c} g^{-1}\xi(i) \tag{11}$$

2 The fixPredict package

2.1 Installing and loading the package

2.2 Example 1: Biased and unbiased model predictions

Let us generate data for a simple two-level Poisson model. The model assumes that a hypothetical variable Y depends on the two covariates X1 and X2, their interaction, and identification number ID. The latter variable is modeled as random intercept. To generate the model we use the `sim_glmer_data` function (see appendix for the code). We choose big, 1000-individual population to minimize the sampling error.

```
# generate a data
set.seed(5)
data1 <- sim_glmer_data(formula = ~ X1 * X2, theta = 0.75,
                        coef= c(5, 0.5, 0.7, 0.5, -0.7, 0.2, 0.3, 0.3, 0.2),
                        n.levels=c(3,3), n.ID = 100, n.pop=1e3)
# show the data
head(data1)
```

```
     Y    linkY X1 X2  ID       IDv
1 1349 7.183494  c bb ID1 0.6834944
2  332 5.783494  b cc ID1 0.6834944
3  285 5.683494  a aa ID1 0.6834944
```

```
4  350 5.883494  c cc ID1 0.6834944
5 1297 7.183494  c bb ID1 0.6834944
6  158 4.983494  a cc ID1 0.6834944
```

where `Y` is a count response, `logY` is its logarithm (latent scale), `X1` and `X2` are factors, `ID` is a identifier of an individual, and `IDv` is a contribution of individual to the intercept on the latent scale. `logY` and `IDv` have only informative property - they are not used to fit models. Knowing the formula that generated the data, we can use it to fit `glmer` model.

```
# fit the model
fit1 <- lme4::glmer(Y~X1*X2+(1|ID),family=poisson(),data=data1)

# comapre the data-generating coeficients and the fitted coeficients
rbind(cbind(data=attr(data1,'coef'),
            fit=round(fixef(fit1),2)),
      theta=round(
          c(data=attr(data1,'theta'),
            fit=getME(fit1,'theta')),2))
```

```
             data   fit
(Intercept)  5.00  5.01
X1b          0.50  0.48
X1c          0.70  0.69
X2bb         0.50  0.50
X2cc        -0.70 -0.72
X1b:X2bb     0.20  0.20
X1c:X2bb     0.30  0.30
X1b:X2cc     0.30  0.31
X1c:X2cc     0.20  0.21
theta        0.75  0.76
```

It is clear, that model precisely recaptured the coefficients used to generate the data. We can now perform model predictions using the generic `predict` function (or the `lme4:::predict.merMod` equivalently).

```
# calculate and plot data (means)
DATA <- with(data1, tapply(Y, data.frame(X1=X1,X2=X2), mean))
posX <- as.vector(barplot(DATA,beside=TRUE,ylim=c(0,1050),
                          ylab='Counts (response scale)'))
text(posX,-40,rep(rownames(DATA),3),xpd=TRUE);
axis(1,at=c(-5,100),labels = FALSE)

# calculate the model predictions
newdata <- expand.grid(X1=levels(data1$X1), X2=levels(data1$X2))
posY <- predict(fit1,
                newdata=newdata,
                type='response',
                re.form = NA)
# plot the predicted responses
lines(posX,posY,pch=20,type='p',col='red2',cex=1.5)

# plot the legend
legend('topleft',c('Simulated data','Default prediction {lme4}\n of the response'),
       pch=c(NA,19),pt.cex=c(1,1.5),
       fill=c(gray.colors(3)[2],NA),col=c(NA,2),bty='n',border=c('black',NA))
```

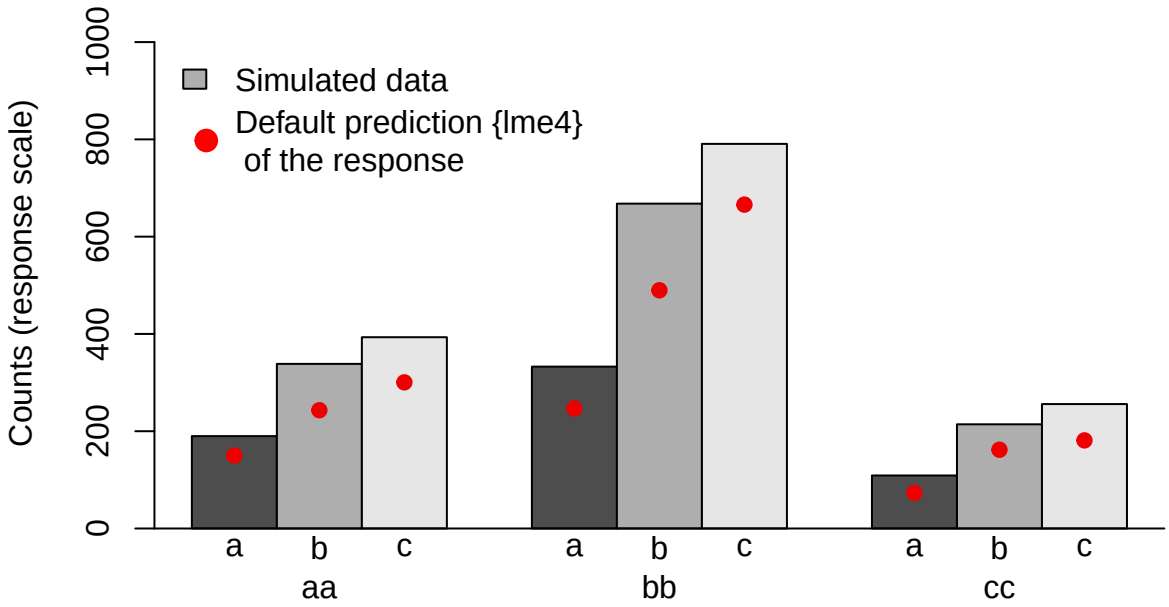


Figure 1. Artificially generated Poisson data and the response predicted by the fitted `glmer` model.

The results show that predictions on the response scale are highly underestimated. It is easy to show that that the bias depends on the magnitude of the theta parameter.

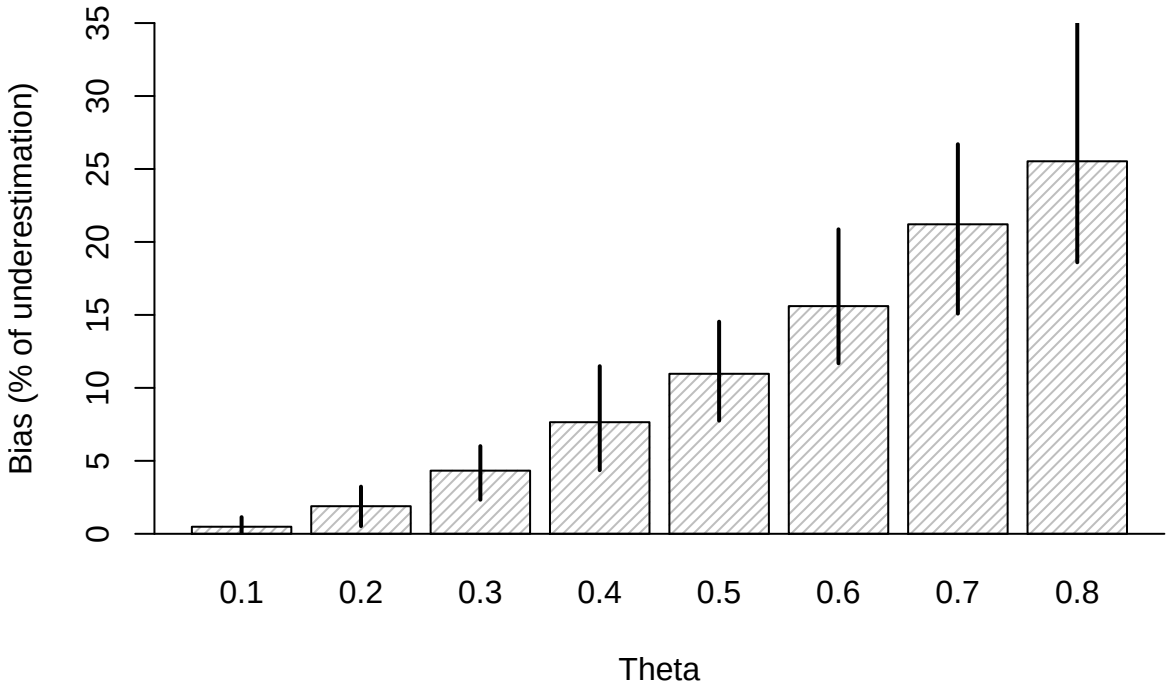
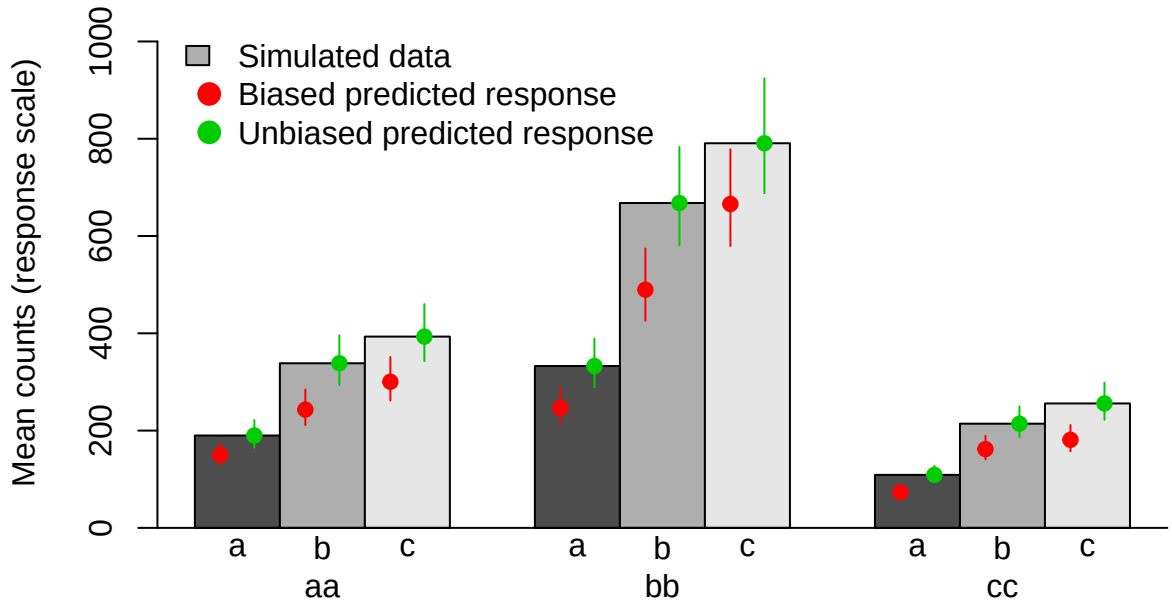


Figure 2. Estimate of the bias between predictions on the response scale and the real data. The confidence intervals were calculated according to the bootstrap percentile method for 100 replicates.

The unbiased predictions on response scale can be achieved using the `fixPredict` function.

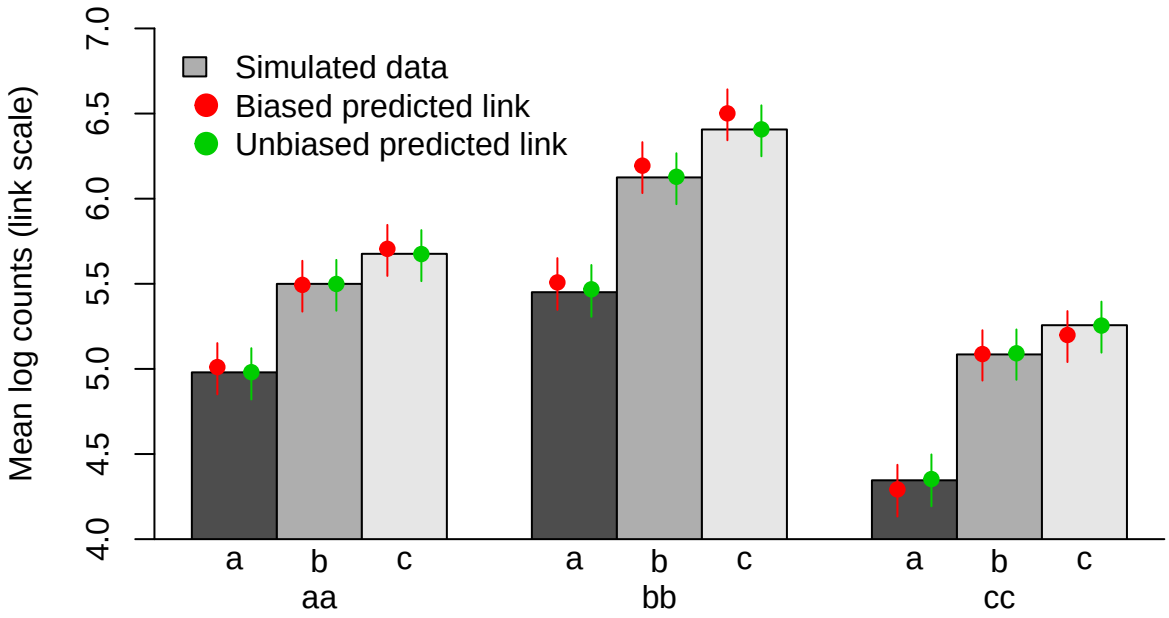
```
# plot the data (means)
posX <- as.vector(barplot(DATA,beside=TRUE,ylim=c(0,1050),
                          ylab='Mean counts (response scale)'))
text(posX,-40,rep(rownames(DATA),3),xpd=TRUE)
```

```
axis(1,at=c(-5,100),labels = FALSE)

# calculate the biased and the unbiased predictions on the response scale
cY <- fixPredict(object=fit1,
                      newdata=as.data.frame(newdata),
                      type='response')

# plot the predicited responses
lines(posX-0.2,cY$fit$biased,pch=20,type='p',col='red',cex=1.5)
lines(posX+0.2,cY$fit$unbiased,pch=20,type='p',col='green3',cex=1.5)

# plot the confidence intervals (the bootstrap percentile method)
for (i in seq_along(posX)) lines(c(posX[i],posX[i])+0.2,
                                 cY$CI.fit$unbiased[i,], col='green3')
for (i in seq_along(posX)) lines(c(posX[i],posX[i])-0.2,
                                 cY$CI.fit$biased[i,], col='red')
legend('topleft',c('Simulated data','Biased predicted response',
                   'Unbiased predicted response'),
       pch=c(NA,19,19),pt.cex=c(1,1.5,1.5),
       fill=c(gray.colors(3)[2],NA,NA),
       col=c(NA,2,'green3'), bty='n', border=c('black',NA,NA))
```



It must be noted that the individual contributions to the random effects are assumed to be fixed and are not drawn during the bootstrap method.

The bias is visible also on the link (latent) scale, however the magnitude is much smaller



2.3 Example 2: GAMM

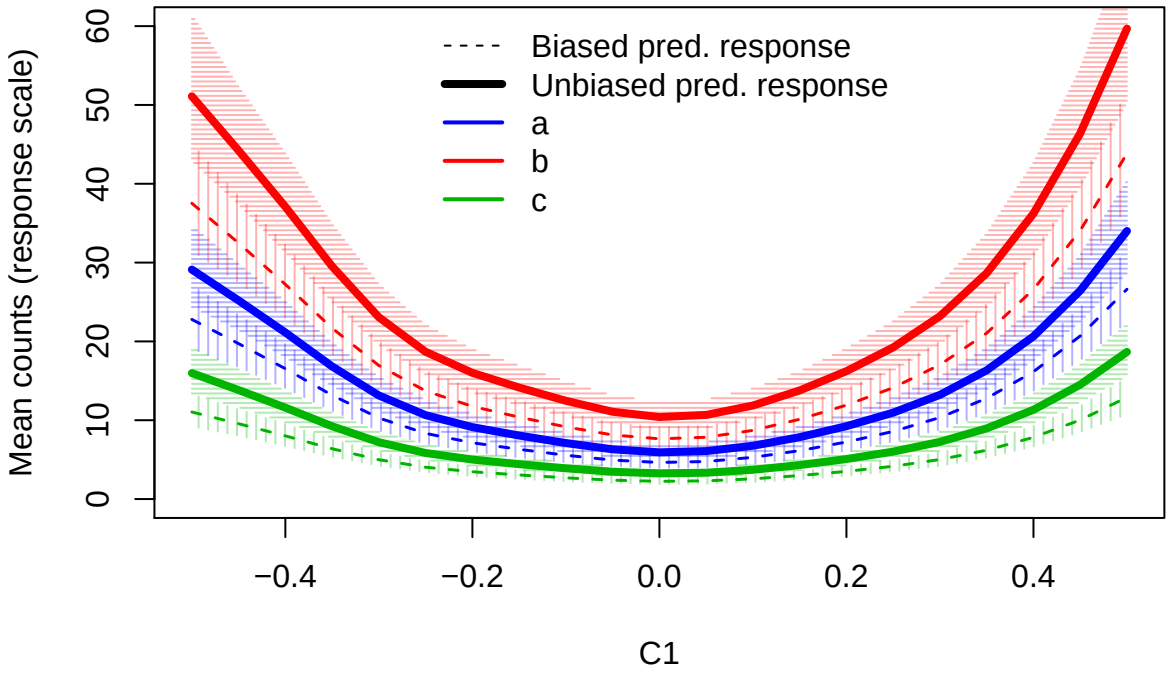
The **fixPredict** package supports two GAMM related packages **mgcv** and **gamm4** Let generate a Poisson data with one categorical and one continouse variable.

```
set.seed(5)
data2 <- sim_glmer_data(formula = ~ X1, theta = 0.75,
                        coef= c(3, 0.5, -0.7),
                        coef.c = 5,
                        func.c = function(C1, coef.c)
                          abs(C1)^1.5 * coef.c - 1.5,
                        n.levels=3, n.ID = 100, n.pop=1e3)
```

The artificialy generated `data2` can be fitted using the `gamm4` function of the **gamm4** package.

```
# fitting the model
fit2 <- gamm4::gamm4(Y~ X1 + s(C1), random = ~(1|ID),
                     family = poisson(), data = data2)

# making the predictions
newdata2 <- expand.grid(C1=seq(-0.5,0.5,0.05), X1=levels(data2$X1))
cY2 <- fixPredict(fit2, newdata2, type = 'response', ci.fit = TRUE)

# plotting the results
ina <- which(newdata2$X1=='a'); inb <- which(newdata2$X1=='b')
inc <- which(newdata2$X1=='c')
xa <- newdata2$C1[ina]; xb <- newdata2$C1[inb]; xc <- newdata2$C1[inc]

CIplot.ci(xa, cY2$fit$unbiased[ina], cY2$CI.fit$unbiased[ina,],
          density=40, angle=0,first = TRUE,ylim=c(0,60),col='blue',lwd=4,
          xaxt='s',yaxt='s', ylab='Mean counts (response scale)', xlab='C1')
CIplot.ci(xa, cY2$fit$unbiased[inb], cY2$CI.fit$unbiased[inb,],
          density=40, angle=0,col='red',lwd=4)
```

```
CIplot.ci(xa, cY2$fit$unbiased[inc], cY2$CI.fit$unbiased[inc,],
          density=40, angle=0,col=rgb(0,180,0,maxColorValue = 255),lwd=4)
CIplot.ci(xa, cY2$fit$biased[ina], cY2$CI.fit$biased[ina,],
          density=20, angle=90,col='blue',lty=2)
CIplot.ci(xa, cY2$fit$biased[inb], cY2$CI.fit$biased[inb,],
          density=20, angle=90,col='red',lty=2)
CIplot.ci(xa, cY2$fit$biased[inc], cY2$CI.fit$biased[inc,],
          density=20, angle=90,col=rgb(0,180,0,maxColorValue = 255),lty=2)

legend('top',c('Biased pred. response', 'Unbiased pred. response',
               'a','b','c'), lty=c(2,1,1,1,1), lwd=c(1,4,2,2,2),
       col=c(1,1,'blue','red', rgb(0,180,0,maxColorValue = 255)), bty='n')
```

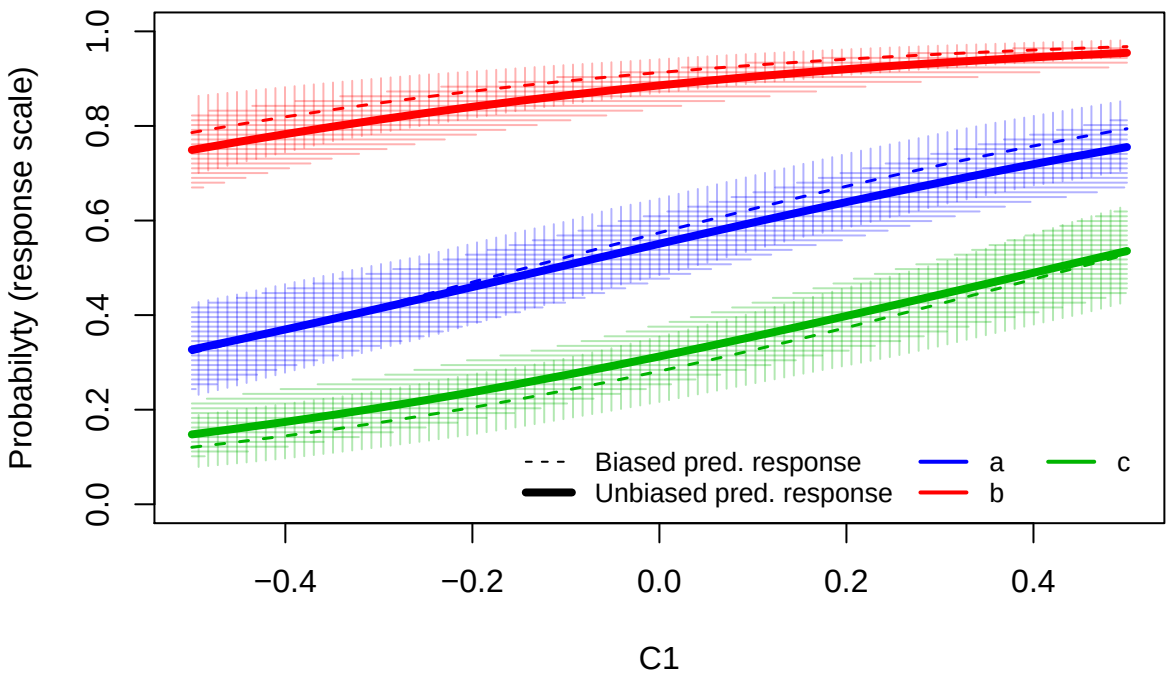


2.4 Example 3: Multi-level logistic regression

```
set.seed(5)
data3 <- sim_glmer_data(formula = ~ X1, theta = 0.9,
                        coef= c(0.5, 1.5, -1.25),
                        coef.c = 2,
                        func.c = function(C1, coef.c) C1 * coef.c,
                        n.levels=3, n.ID = 100, n.pop=1e3,
                        family='binomial')
```

```
# fitting the model
fit3 <- glmer(Y ~ X1 + C1 + (1|ID), family=binomial(), data=data3, nAGQ = 20)

# making the predictions
newdata3 <- expand.grid(C1=seq(-0.5,0.5,0.05), X1=levels(data3$X1))
cY3 <- fixPredict(fit3, newdata3, n.sim = 1000,
```

```
                  type = 'response', ci.fit = TRUE)

# plotting the results
ina <- which(newdata3$X1=='a'); inb <- which(newdata3$X1=='b')
inc <- which(newdata3$X1=='c')
xa <- newdata3$C1[ina]; xb <- newdata3$C1[inb]; xc <- newdata3$C1[inc]

CIplot.ci(xa, cY3$fit$unbiased[ina], cY3$CI.fit$unbiased[ina,],
          density=40, angle=0,first = TRUE,ylim=c(0,1),col='blue',lwd=4,
          xaxt='s', yaxt='s', ylab='Probabilyty (response scale)', xlab='C1')
CIplot.ci(xa, cY3$fit$unbiased[inb], cY3$CI.fit$unbiased[inb,],
          density=40, angle=0,col='red',lwd=4)
CIplot.ci(xa, cY3$fit$unbiased[inc], cY3$CI.fit$unbiased[inc,],
          density=40, angle=0,col=rgb(0,180,0,maxColorValue = 255),lwd=4)
CIplot.ci(xa, cY3$fit$biased[ina], cY3$CI.fit$biased[ina,],
          density=20, angle=90,col='blue',lty=2)
CIplot.ci(xa, cY3$fit$biased[inb], cY3$CI.fit$biased[inb,],
          density=20, angle=90,col='red',lty=2)
CIplot.ci(xa, cY3$fit$biased[inc], cY3$CI.fit$biased[inc,],
          density=20, angle=90,col=rgb(0,180,0,maxColorValue = 255),lty=2)

legend('bottomright',c('Biased pred. response', 'Unbiased pred. response',
                       'a','b','c',''), lty=c(2,1,1,1,1,NA), lwd=c(1,4,2,2,2,NA),
       col = c(1,1,'blue','red', rgb(0,180,0,maxColorValue = 255),NA), bty='n',
       cex = 0.8, text.width = strwidth('a')*c(7,7,2,2,2,NA), ncol = 3)
```

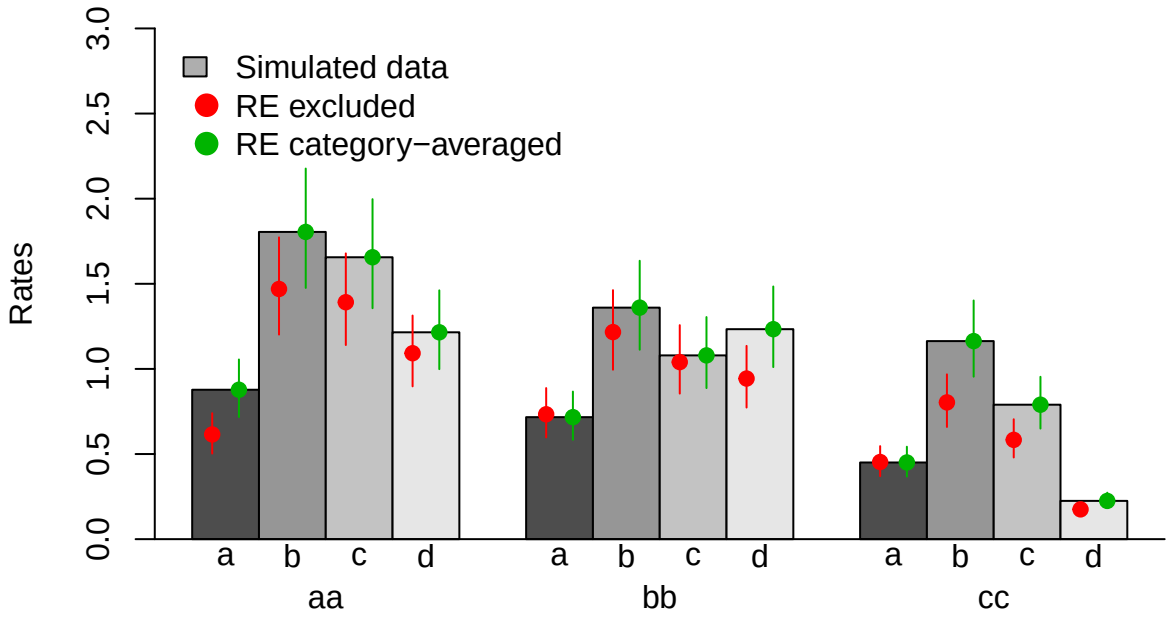


The data generated in this example has assumed very high `theta` parameter. It is thus clear that binomial logistic regression is more resistant to the bias resulting from ignoring random effects.

2.5 Example 4: Count data with offset and rates

```
set.seed(5)
data4 <- sim_glmer_data(formula = ~ X1 * X2, theta = 0.75,
                        coef= c(5, 0.9, 0.7, 0.4,-0.2, -0.7, 0, 0, 0, 0, 0,-1.1),
                        n.levels=c(4,3), n.ID = 100, n.pop=1e3)
# generating random exposures
data4$exposures <- exp(rnorm(1e3,5))
```

```
# fit the model
fit4 <- lme4::glmer(Y~X1*X2+(1|ID), family=poisson(),
                    offset=log(data4$exposures), data=data4, nAGQ = 20)

newdata4 <- expand.grid(X1=levels(data4$X1), X2=levels(data4$X2))

# calculate the biased and the unbiased predictions on the response scale
cY4 <- fixPredict(object = fit4,
                  newdata = newdata4,
                  type = 'response',
                  use.offset = TRUE,
                  as.rate = TRUE,
                  ci.fit = TRUE,
                  method = 'at.each.cat')

DATA.counts <- with(data4, tapply(Y, data.frame(X1=X1, X2=X2), sum))
DATA.exposures <- with(data4, tapply(exposures, data.frame(X1=X1,X2=X2), sum))
DATA.rates <- DATA.counts/DATA.exposures

posX <- as.vector(barplot(DATA.rates,beside=TRUE, ylim=c(0,3),
                          ylab='Rates'))
text(posX,-0.1,rep(rownames(DATA.counts),3),xpd=TRUE)
axis(1,at=c(-5,100),labels = FALSE)

lines(posX-0.2,cY4$fit$biased,pch=20,type='p',col='red',cex=1.5)
lines(posX+0.2,cY4$fit$unbiased,pch=20,type='p',
      col=rgb(0,180,0,maxColorValue = 255),cex=1.5)

# plot the confidence intervals (the bootstrap percentile method)
for (i in seq_along(posX)) lines(c(posX[i],posX[i])+0.2,
                                 cY4$CI.fit$unbiased[i,],
                                 col=rgb(0,180,0,maxColorValue = 255))
for (i in seq_along(posX)) lines(c(posX[i],posX[i])-0.2,
                                 cY4$CI.fit$biased[i,], col='red')

legend('topleft',c('Simulated data','RE excluded',
                   'RE category-averaged'),
       pch=c(NA,19,19),pt.cex=c(1,1.5,1.5),
       fill=c(gray.colors(3)[2],NA,NA),
       col=c(NA,2,rgb(0,180,0,maxColorValue = 255)),
       bty='n', border=c('black',NA,NA))
```



2.6 Example 5: Predicitions with vriables missing in the `newdata`

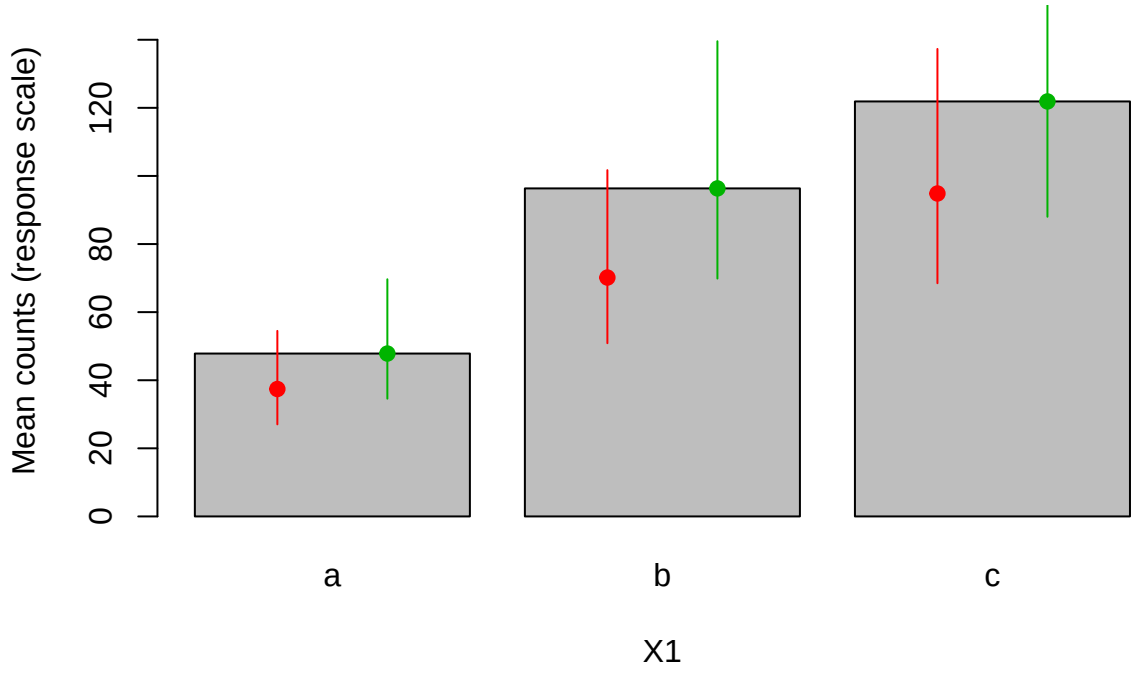
```
set.seed(5)
data6 <- sim_glmer_data(formula = ~ X1 * X2, theta = 0.75,
                        coef= c(3.5, 0.5, 0.7, 0.5, -0.7, 0.2, 0.3, 0.3, 0.2),
                        coef.c = -1.5,
                        n.levels=c(3,3), n.ID = 20, n.pop=1e3)

fit6 <- lme4::glmer(Y~X1*X2+C1+(1|ID),family=poisson(), data=data6)

DATA <- with(data6, tapply(Y, data.frame(X1=X1), mean))
```

```
newdata6 <- expand.grid(X1=levels(data6$X1))

# error due to missing variables in newdata
cY6 <- try(fixPredict(object=fit6,
                      newdata = newdata6,
                      type='response',
                      ci.fit = TRUE,
                      method = 'at.each.cat'))
```

```
Error : Missing variable(s) in newdata: X2, C1.
 Add missing variables or set average.missing = TRUE.
```

```
# averaging missing variables
cY6 <- try(fixPredict(object = fit6,
                      newdata = newdata6,
                      type='response',
                      average.missing = TRUE,
                      ci.fit = TRUE,
                      method = 'at.each.cat'))
```

```
posX <- as.vector(barplot(DATA,beside=TRUE,ylim=c(0,150),
                          ylab='Mean counts (response scale)',
                          xlab='X1'))
lines(posX-0.2,cY6$fit$biased, type='p',col='red',pch=20,cex=1.5)
lines(posX+0.2,cY6$fit$unbiased,pch=20,type='p',col=rgb(0,180,0,maxColorValue = 255),cex=1.5)
for (i in seq_along(posX)) lines(c(posX[i],posX[i])+0.2,
                                 cY6$CI.fit$unbiased[i,], col=rgb(0,180,0,maxColorValue = 255))
for (i in seq_along(posX)) lines(c(posX[i],posX[i])-0.2,
                                 cY6$CI.fit$biased[i,], col='red')
```



3 Conclusions

4 References