Likelihood analysis for a class of beta mixed models

Supplement material: The algorithm

Wagner Hugo Bonat [*], Paulo Justiniano Ribeiro Jr,

Walmes Marques Zeviani

LEG/DEST - Paraná Federal University

Abstract

The main goal this supplement material is describes the algorithm used to fit beta mixed models. Our algorithm is based on Laplace approximation and numerical maximization method BFGS. The integrand of likelihood function is a product between the beta and Gaussian distributions. We use Laplace approximation to integrate out the random effects and BFGS method to find the maximum of approximated log-likelihood function. In the following we present a brief code to explain the main steps about the implementation of beta mixed models.

keywords: R, likelihood inference, Laplace approximation, beta, random effects

1 Beta mixed models - Brief code

In this report we present the main steps to implement the beta mixed model. All functions are programmed in `R`. This report was done to be self-sufficient and describes a complete analysis using beta mixed models.

[*]Corresponding author: wagner@ufpr.br, Dept. Estatística-UFPR, CP 19.081, Curitiba, PR Brazil, 81.531-990

We will use simulated data and unidimensional random effects. The code shows a generic function to simulate from a beta mixed model. The simulated model has only three parameters $\theta = (\beta, \phi, \tau)$, where β is an intercept, ϕ is the dispersion parameter from beta law and τ is the precision of the random effect. We opted to simulate 10 unit samples every with 10 observations.

```
> simula.beta <- function(para, n.bloco=10, n.rep=5){
+                bloco <- rep(1:n.bloco,each=n.rep)
+                bloco.efeito <- rep(rnorm(n.bloco,0, sd=1/para[3]),each=n.rep)
+                eta <- para[1] + bloco.efeito
+                mu <- exp(eta)/(1+exp(eta))
+                y <- rbeta(length(mu), mu*para[2], (1-mu)*para[2])
+                return(data.frame(y=y, ID= bloco))
+ }
```

Simulating the data set.

```
> set.seed(123)
> dados <- simula.beta(para=c(0.5, 50, 10), n.bloco=10, n.rep = 10)
```

The next step is to build a function to compute the Laplace approximation.

```
> laplace <- function(funcao, otim, n.dim, ...){
+            integral <- -sqrt(.Machine$double.xmax)
+            inicial <- rep(0,n.dim)
+            temp <- try(optim(inicial,funcao, ...,
+            method=otim, hessian=TRUE,control=list(fnscale=-1)))
+            if(class(temp) != "try-error"){
+                integral <- exp(temp$value)* (exp(n.dim*log(2*pi) -
+                          0.5*determinant(-temp$hessian)$modulus))}
+   return(integral)
+ }
```

To make easy we implement a generic function to evaluate the log-likelihood function. To use this function we need to implement the model in a suitable way (describe below). Arguments are, `formu.X` design matrix of fix effects, `formu.Z` design matrix of random effects. The values for β and precision parameters correspond the arguments `beta.fixo` and `prec.pars` respectivelly. Extra argument are `otim` the optimization method, `n.dim` the dimension of random effects structure and `data` the data set.

```
> likelihood <- function(modelo, formu.X, formu.Z, beta.fixo,
+                         prec.pars, otim, n.dim, data){
+               dados.id <- split(data, data$ID)
+               ll <- c()
+           for(i in 1:length(dados.id)){
+                  X <- model.matrix(as.formula(formu.X),
+                                    data=dados.id[[i]])
+              Z <- model.matrix(as.formula(formu.Z),
+                                    data=dados.id[[i]])
+     ll[i] <- laplace(modelo, otim=otim, n.dim=n.dim,
+                          X=X, Z=Z , Y=dados.id[[i]]$y,
+                          beta.fixo=beta.fixo,
+                          prec.pars=prec.pars,log=TRUE)
+ }
+ return(sum(log(ll)))
+ }
```

Inverse of logit link function.

```
> inv.logit <- function(x){exp(x)/(1+exp(x))}
```

Next step is written the model in a suitable way to use inside the function `likelihood`, the following code show how to do it.

```
> model.fit <- function(uv, beta.fixo, prec.pars, X, Z, Y,log=TRUE){
+   phi <- exp(prec.pars[1])
+   tau <- exp(prec.pars[2])
+   if(class(dim(uv)) == "NULL"){uv <- matrix(uv,1,length(uv))}
+   ll = apply(uv,1,function(uvi){
+     preditor <- X%*%beta.fixo + Z%*%as.numeric(uvi)
+     mu <- inv.logit(preditor)
+     sum(dbeta(Y, mu*phi, (1-mu)*phi, log=TRUE)) + dnorm(uvi[1], 0, sd = 1/tau , log=
+   if(log == FALSE){ll <- exp(ll)}
+   return(ll)}
```

Now, we have enough information to compute the likelihood.

```
> dados$intercepto <- 1
> modelo.ajuste <- function(b1,phi,tau,n.dim,otim, data){
+                   ll = likelihood(modelo = model.fit,
+                        formu.X="~ intercepto -1 ",
+                        formu.Z="~ intercepto -1",
+                        beta.fixo = b1, prec.pars=c(phi,tau),
+                        otim = otim, n.dim = n.dim,data=data)
+   #print(round(c(b1,phi,tau,ll),2))
+   return(-ll)}
```

Find the maximum likelihood estimates using the `BFGS` algorithm from `optim` function using the package `bbmle`.

```
> ajuste.1 = mle2(modelo.ajuste,
+                   start=list(b1=0,phi=log(1), tau=log(1)),
+                   data=list(otim ="BFGS", n.dim=1, data= dados))
```

Summary of point estimates and asymptotic confidence interval.

```
> summary(ajuste.1)
> confint(ajuste.1,method="quad")
```

Profile likelihood.

```
> perfil.beta <- profile(ajuste.1)
```

It was a brief implementation of beta mixed models using Laplace approximation.