CE-227: Inferência Bayesiana – 4^a Avaliação Semanal (16/04/2014)

GRR: ________________ **Nome:** ____ **Turma:** _________

Descreva o modelo sendo ajustado e a estrutura dos dados nas seguintes declarações de modelos em JAGS.

1.
```
model{
  for (i in 1:N){
    x[i] ~ dbern(p)
  }
  p ~ dbeta(alpha, beta)
  alpha <- 1
  beta <- 1
}
```

Modelo binomial (= ensaios Bernoulli independentes):

$$\begin{aligned}
&\text{verossimilhança:} \\
&\quad X_i|p \sim \text{Bern}(p_i) \\
&\text{priori:} \\
&\quad p_i \sim \text{Be}(\alpha = 1, \beta = 1)(\equiv \text{U}[0,1])
\end{aligned}$$

Simulação de dados do modelo

```
> set.seed(227)
> p <- runif(1, 0, 1)
> n <- 30
> x <- rbinom(30, size=1, prob=p)
> #Elias (define the list data here instead later, like in ex2)
> q1.dat <- list(x = x, N = length(x))
```

- Estimação não-Bayesiana
  i. Solução analítica (MLE) é disponível neste caso

$$\hat{p} = \frac{\sum_i x_i}{n} = 0.7$$
$$\text{sd}(\hat{p}) = \sqrt{1/I_o(\hat{p})} = \sqrt{\hat{p}(1-\hat{p})/n} = 0.0837$$

  ii. (Algumas) soluções numéricas

```
> qa.glm <- glm(x ~ 1, family=binomial(link="logit"))
> unlist(predict(qa.glm, newdata = data.frame(c(1)), type="response", se.fit=TRUE)[1:2])
   fit.1 se.fit.1
 0.70000  0.08367
> #
> lik <- function(p, data, log=TRUE){
+     dbinom(sum(data), size=length(data), prob=p, log=log)
+ }
> (qa.optim <- optimize(lik, interval=c(0,1), data=x, maximum=TRUE))
$maximum
[1] 0.7

$objective
[1] -1.85

> with(qa.optim, sqrt(-1/drop(optimHess(par=maximum, fn=lik, data=x))))
[1] 0.08366
```

- Estimação Bayesiana

i. Posteriori analítica (conjugada neste caso)

$$p|x \sim \text{Be}(\alpha^* = \alpha + \sum_i x_i = 1 + 21 = 22, \beta^* = \beta + n - \sum_i x_i = 1 + 30 - 21 = 10)$$

$$\text{E}[p|x] = \frac{\alpha^*}{\alpha^* + \beta^*} = \frac{\alpha + \sum_i x_i}{\alpha + \beta + n} = 0.688$$

$$\text{Mo}[p|x] = \frac{\alpha^* - 1}{\alpha^* + \beta^* - 2} = \frac{\alpha + \sum_i x_i - 1}{\alpha + \beta + n - 2} = 0.7$$

$$\text{Var}[p|x] = \frac{\alpha^* \beta^*}{(\alpha^* + \beta^*)^2(\alpha^* + \beta^* + 1)} = 0.00651$$

$$\text{SD}[p|x] = 0.0807$$

ii. Amostras (MCMC) - JAGS

```
> require(rjags)
> cat("model{
+   for (i in 1:N){
+     x[i] ~ dbern(p)
+   }
+   p ~ dbeta(alpha, beta)
+   alpha <- 1
+   beta <- 1
+   }", file="av04-q1.jags")
> inis <- list(list(p=0.1), list(p=0.5), list(p=0.9))
> q1.model <- jags.model(file="av04-q1.jags", data=q1.dat, n.chains=3, init = inis)
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 34

Initializing model
```

```
> q1.sam <- coda.samples(q1.model, c("p"), 20000, thin=10)
> summary(q1.sam)
```

```
Iterations = 10:20000
Thinning interval = 10
Number of chains = 3
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean             SD       Naive SE Time-series SE
       0.68708        0.08219        0.00106        0.00106

2. Quantiles for each variable:

 2.5%   25%   50%   75% 97.5%
0.517 0.634 0.691 0.746 0.834
```

iii. INLA

```
> require(INLA)
> q1.inla <- inla(x ~ 1, data=q1.dat, family="binomial", control.family=list(link="logit"),
+                   control.predictor=list(compute=TRUE))
> summary(q1.inla)
```

```
Call:
c("inla(formula = x ~ 1, family = \"binomial\", data = q1.dat, control.predictor = list(compute = TRU

Time used:
 Pre-processing    Running inla Post-processing           Total
         0.8259          0.7382          0.1229          1.6870

Fixed effects:
              mean     sd 0.025quant 0.5quant 0.975quant   mode kld
(Intercept) 0.8473 0.3984     0.0972   0.8354      1.664 0.8112   0
```

```
The model has no random effects

The model has no hyperparameters

Expected number of effective parameters(std dev): 1.00(0.00)
Number of equivalent replicates : 30.00

Marginal Likelihood:  -18.33
Posterior marginals for linear predictor and fitted values computed
```

```
> q1.inla$summary.fitted.values[1,]
                       mean      sd 0.025quant 0.5quant 0.975quant   mode
fitted.predictor.01 0.6936 0.08143     0.5245   0.6975     0.8407 0.7057
> inla.zmarginal(q1.inla$marginals.fitted.values[[1]])
Mean            0.693629
Stdev           0.081431
Quantile  0.025 0.524297
Quantile  0.25  0.639414
Quantile  0.5   0.697183
Quantile  0.75  0.751411
Quantile  0.975 0.840348
```

O modelo ajustado pelo INLA não corresponde exatamente ao do JAGS. No INLA, os coeficientes de regressão são tratados da mesma forma que os efeitos aleatórios. Então, a priori é necessariamente gaussiana (no caso, foi usada a priori vaga, opção *default*), pois o algoritmo é baseado nisso. O modelo ajustado pelo INLA é então: Modelo binomial (= ensaios Bernoulli independentes):

verossimilhança:

$$X_i|p \sim \text{Bern}(p_i)$$

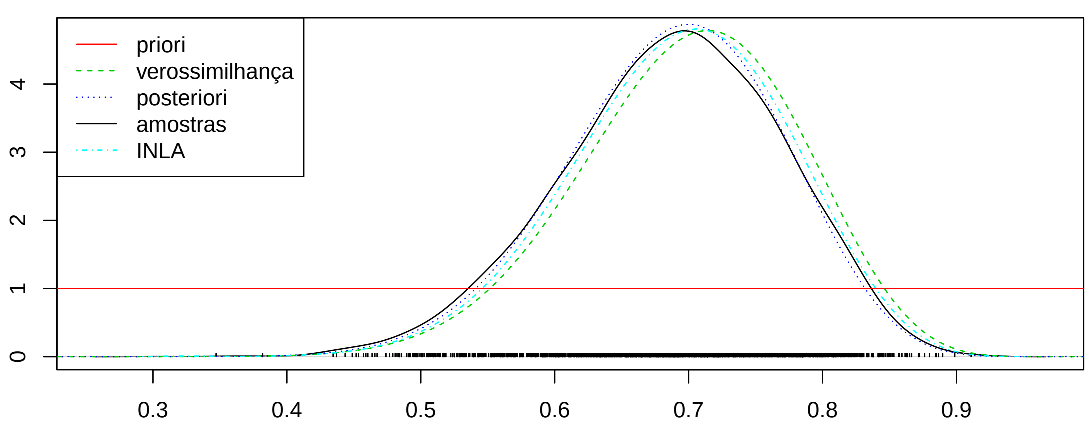
$$\log\left(\frac{p}{1-p}\right) = \beta_0 \text{priori:}$$

$$\beta_0 \sim \text{N}(0, +\infty)(\equiv \text{U}[-\infty, +\infty])$$

Fica como exercício reescrever o código JAGS que corresponda ao modelo ajustado pelo INLA.

Gráficos da priori, verossimilhança e posteriori

```
> p.vals <- seq(0,1,l=501)
> pr.vals <- cbind(dbeta(p.vals, 1, 1), dbeta(p.vals, sum(x), length(x)-sum(x)),dbeta(p.vals, a1, b1))
> densplot(q1.sam, main="", xlab="")
> matlines(p.vals, pr.vals, type="l", xlab="p", ylab="", col=2:4)
> lines(spline(dp.temp ~ p.temp), col=5, lty=4)
> legend("topleft", c("priori","verossimilhança","posteriori", "amostras", "INLA"), lty=c(1:3,1,4), col=c
```



2. model{

```
model{
   for(i in 1:M){
      for(j in 1:N){
         y[i,j] ~ dnorm(mu[i], tau)
      }
      mu[i] ~ dnorm(theta, tauD)
   }
   tau <- pow(sigma, -2)
   sigma ~ dunif(0, 100)
   theta ~ dnorm(0, .001)
   tauD <- pow(delta, -2)
   delta ~ dunif(0, 100)
}
```

Modelo (medidas repetidas):

$$\begin{aligned} &\text{verossimilhança:} \\ &Y_{ij}|\mu_i, \sigma^2 \sim \mathrm{N}(\mu_i, \sigma^2) \\ &\text{variáveis latentes (ef. al.):} \\ &\mu_i \sim \mathrm{N}(\theta, \delta^2) \\ &\text{priori's:} \\ &\theta \sim \mathrm{N}(0, 1000) \\ &\sigma \sim \mathrm{U}[0, 100] \\ &\delta \sim \mathrm{U}[0, 100] \end{aligned}$$

Simulação de dados do modelo proposto

```
> set.seed(22701)
> Ng <- 10
> Nobs <- 5
> N <- Ng*Nobs
> delta <- 5
> sigma <- 2
> mus <- rnorm(Ng, mean=50, sd=delta)
> y <- matrix(rnorm(N, mean=mus, sd=sigma), ncol=Ng, byrow=TRUE)
> q2.df <- data.frame(y = as.vector(y), grupo = rep(1:Ng, each=Nobs))
```

- Estimação não-Bayesiana

```
> require(lme4)
> (q2.lmer <- lmer(y ~ 1|grupo, data=q2.df, REML=FALSE))
```

```
Linear mixed model fit by maximum likelihood  ['lmerMod']
Formula: y ~ 1 | grupo
   Data: q2.df
     AIC      BIC   logLik deviance df.resid
   261.3    267.0   -127.6    255.3       47
Random effects:
 Groups   Name        Std.Dev.
 grupo    (Intercept) 3.90
 Residual             2.38
Number of obs: 50, groups: grupo, 10
Fixed Effects:
(Intercept)
       49.6
```

```
> t(ranef(q2.lmer)$grupo)
```

```
                 1      2     3     4      5     6      7      8     9    10
(Intercept) -3.104 -0.207 1.287 5.898 -4.738 2.586 -4.984 -3.535 1.693 5.104
```

```
> t(ranef(q2.lmer)$grupo +  fixef(q2.lmer)[[1]])
```

```
                1     2     3     4     5     6     7     8     9    10
(Intercept) 46.46 49.36 50.85 55.47 44.83 52.15 44.58 46.03 51.26 54.67
```

- Estimação Bayesiana
  - JAGS

```
> require(rjags)
> cat("model{
+   for(i in 1:M){
+      for(j in 1:N){
+         y[i,j] ~ dnorm(mu[i], tau)
+      }
+      mu[i] ~ dnorm(theta, tauD)
+   }
+   tau <- pow(sigma, -2)
+   sigma ~ dunif(0, 100)
+   theta ~ dnorm(0, .001)
+   tauD <- pow(delta, -2)
+   delta ~ dunif(0, 100)
+ }", file="av04-q2.jags")
> q2.dat <- list(y = t(y), M = Ng, N = Nobs)
> q2.model <- jags.model(file="av04-q2.jags", data=q2.dat, n.chains=3)
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 72

Initializing model
> q2.sam <- coda.samples(q2.model, c("mu", "sigma", "delta", "theta"), 20000, thin=10)
> summary(q2.sam)
Iterations = 1010:21000
Thinning interval = 10
Number of chains = 3
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

        Mean    SD Naive SE Time-series SE
delta   4.85 1.511  0.01951        0.02002
mu[1]  46.41 1.092  0.01410        0.01448
mu[2]  49.33 1.072  0.01384        0.01371
mu[3]  50.85 1.078  0.01391        0.01391
mu[4]  55.52 1.100  0.01420        0.01439
mu[5]  44.77 1.105  0.01427        0.01405
mu[6]  52.17 1.075  0.01387        0.01412
mu[7]  44.53 1.083  0.01398        0.01427
mu[8]  45.99 1.085  0.01401        0.01447
mu[9]  51.29 1.070  0.01381        0.01345
mu[10] 54.73 1.088  0.01405        0.01350
sigma   2.46 0.286  0.00369        0.00369
theta  49.47 1.651  0.02132        0.02251

2. Quantiles for each variable:

        2.5%   25%   50%   75% 97.5%
delta   2.82  3.82  4.57  5.53  8.62
mu[1]  44.28 45.69 46.41 47.10 48.62
mu[2]  47.21 48.62 49.35 50.04 51.44
mu[3]  48.72 50.14 50.85 51.57 52.99
mu[4]  53.33 54.78 55.52 56.25 57.69
mu[5]  42.59 44.03 44.77 45.50 46.98
mu[6]  50.05 51.47 52.17 52.86 54.24
mu[7]  42.41 43.81 44.52 45.26 46.67
mu[8]  43.86 45.25 45.98 46.70 48.15
mu[9]  49.20 50.58 51.28 52.01 53.40
mu[10] 52.57 54.01 54.73 55.47 56.83
sigma   1.98  2.26  2.43  2.64  3.10
theta  46.10 48.45 49.48 50.48 52.79
```

– INLA

```
> require(INLA)
> q2.inla <- inla(y ~  f(grupo, model="iid"), family="gaussian", data=q2.df)
> summary(q2.inla)
Call:
c("inla(formula = y ~ f(grupo, model = \"iid\"), family = \"gaussian\", ",  "    data = q2.df)")

Time used:
 Pre-processing    Running inla Post-processing           Total
         2.4032          0.7762          0.5593          3.7387

Fixed effects:
             mean    sd 0.025quant 0.5quant 0.975quant  mode kld
(Intercept) 49.57 1.282         47    49.57      52.13 49.57   0

Random effects:
Name          Model
 grupo   IID model

Model hyperparameters:
                                        mean   sd     0.025quant 0.5quant 0.975quant
Precision for the Gaussian observations 0.1830 0.0409 0.1158     0.1786   0.2757
Precision for grupo                     0.0773 0.0365 0.0266     0.0708   0.1665
                                        mode
Precision for the Gaussian observations 0.1701
Precision for grupo                     0.0578

Expected number of effective parameters(std dev): 9.257(0.3401)
Number of equivalent replicates : 5.401

Marginal Likelihood:  -151.38
> ## passando de precisão para variância:
> sqrt(1/q2.inla$summary.hyperpar[,1])
Precision for the Gaussian observations                     Precision for grupo
                                  2.337                                   3.596
```

No **INLA** há funções para manipular as *posterior marginal distribution's (PMDs)* e extrair medidas resumo.

```
> post.sigma = inla.tmarginal(function(x) sqrt(1/x), q2.inla$marginals.hyperpar[[1]])
> post.delta = inla.tmarginal(function(x) sqrt(1/x), q2.inla$marginals.hyperpar[[2]])
> rbind(delta=inla.zmarginal(post.delta, T), sigma=inla.zmarginal(post.sigma, T))
      mean  sd     quant0.025 quant0.25 quant0.5 quant0.75 quant0.975
delta 3.898 0.9337 2.458      3.224     3.754    4.421     6.108
sigma 2.38  0.2608 1.908      2.196     2.365    2.548     2.932
```

Efeitos aleatórios: médias e modas por grupos.

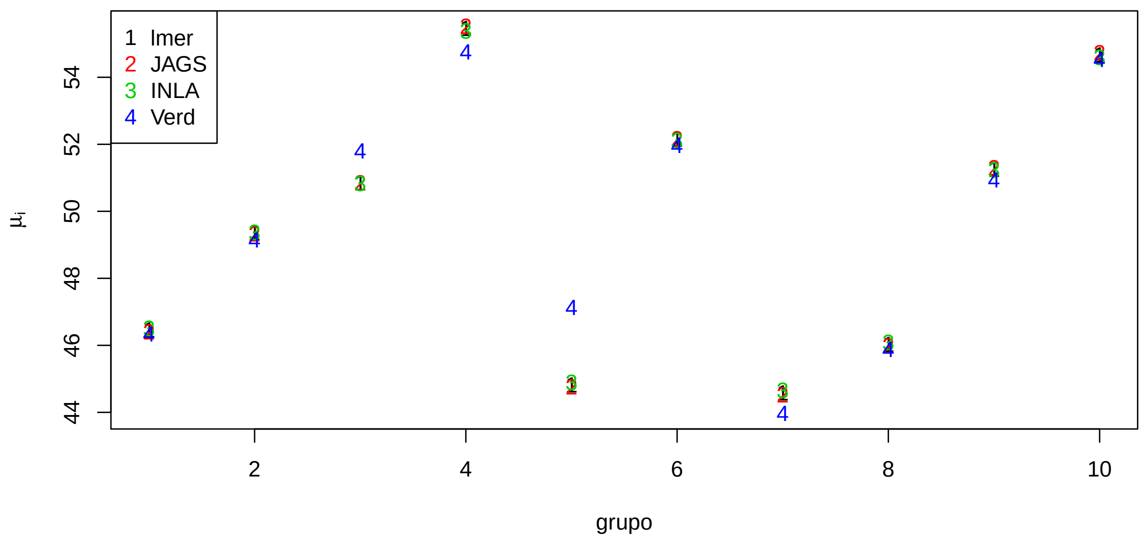
```
> rbind(medias.grupos=q2.inla$summary.fix[1,1] + q2.inla$summary.random$grupo$mean,
+       modas.grupos=q2.inla$summary.fix[1,6] + q2.inla$summary.random$grupo$mode)
               [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10]
medias.grupos 46.51 49.36 50.84 55.38 44.90 52.12 44.65 46.08 51.24 54.60
modas.grupos  46.54 49.37 50.82 55.31 44.95 52.08 44.71 46.13 51.22 54.54
```

Comparação dos valores das variâncias dos grupos e residual.

```
> par(mfrow=c(1,2))
> plot(density(unlist(q2.sam[,"delta",drop=T])), main="", xlab=expression(delta),
+      ylab=expression(group("[",paste(delta,"|",y),"]")), ylim=c(0, 0.5))
> lines(post.delta, lty=2, col=2)
> abline(v=c(as.data.frame(VarCorr(q2.lmer))$sdcor[1], delta), col=3:4, lty=3:4)
> legend("topright", c("JAGS","INLA", "LMER", "Verd."), col=1:4, lty=1:4)
> plot(density(unlist(q2.sam[,"sigma",drop=T])), main="", xlab=expression(sigma),
+      ylab=expression(group("[",paste(sigma,"|",y),"]")), ylim=c(0, 1.5))
> lines(post.sigma, lty=2, col=2)
> abline(v=c(as.data.frame(VarCorr(q2.lmer))$sdcor[2], sigma), col=3:4, lty=3:4)
> legend("topright", c("JAGS","INLA", "LMER", "Verd."), col=1:4, lty=1:4)
```

Médias dos grupos

```
> (q2.grupos <- data.frame(lmer = drop(t(ranef(q2.lmer)$grupo +  fixef(q2.lmer)[[1]])),
+                          JAGS = summary(q2.sam)[[1]][2:11,1],
+                          INLA = q2.inla$summary.fix[1,1] + q2.inla$summary.random$grupo$mean,
+                          verd = mus))
    lmer  JAGS  INLA  verd
1  46.46 46.41 46.51 46.35
2  49.36 49.33 49.36 49.15
3  50.85 50.85 50.84 51.81
4  55.47 55.52 55.38 54.76
5  44.83 44.77 44.90 47.13
6  52.15 52.17 52.12 51.98
7  44.58 44.53 44.65 43.97
8  46.03 45.99 46.08 45.89
9  51.26 51.29 51.24 50.95
10 54.67 54.73 54.60 54.53
> matplot(1:10, q2.grupos, type="p", xlab="grupo", ylab=expression(mu[i]))
> legend("topleft", c("lmer","JAGS","INLA", "Verd"),pch=as.character(1:4), col=1:4)
```



3.
```
model{
  for (i in 1:N){
    y[i] ~ dbern(p[i])
    logit(p[i]) <- a[g[i]] * x[i]
}
for (j in 1:K){
  a[j] ~ dnorm(mu.a, tau.a)
}
mu.a ~ dnorm(0, 0.0001)
tau.a <- pow(sigma.a, -2)
sigma.a ~ dunif(0, 1000)
}
```

Modelo (GLMM - modelo linear generalizado misto):

$$\begin{aligned}
\text{verossimilhança:} & \\
Y_i|p_i &\sim \text{Ber}(p_i) \\
\log\left(\frac{p}{1-p}\right) &= a_i x_i \\
\text{variáveis latentes (ef. al.):} & \\
a_i &\sim \text{N}(\mu_a, \sigma_a^2) \\
\text{priori's:} & \\
\mu_a &\sim \text{N}(0, 10000) \\
\sigma &\sim \text{U}[0, 100]
\end{aligned}$$

Simulação de dados do modelo proposto

```
> set.seed(22701)
> Ng <- 15
> x <- seq(-4, 4, l=15)
> Nx <- length(x)
> N <- Nx*Ng
> as <- rnorm(Ng, mean=0.5, sd=0.2)
> nus <- as.vector(outer(x, as))
> ps <- exp(nus)/(1+exp(nus))
> q3.df <- data.frame(y = rbinom(N, size=1, prob=ps), x = x,  grupo = rep(1:Ng, each=Nx))
```

- Estimação não-Bayesiana

```
> require(lme4)
> q3.glmer <- glmer(y ~ 0 + x + (0+x|grupo), family=binomial, data=q3.df)
> summary(q3.glmer)
```

```
Generalized linear mixed model fit by maximum likelihood (Laplace Approximation) [
glmerMod]
 Family: binomial ( logit )
Formula: y ~ 0 + x + (0 + x | grupo)
   Data: q3.df

     AIC      BIC   logLik deviance df.resid
   256.1    263.0   -126.1    252.1      223

Scaled residuals:
   Min     1Q Median     3Q    Max
-2.474 -0.596  0.356  0.754  3.040

Random effects:
 Groups Name Variance Std.Dev.
 grupo  x    0.0384   0.196
Number of obs: 225, groups: grupo, 15

Fixed effects:
  Estimate Std. Error z value Pr(>|z|)
x   0.5062     0.0951    5.32    1e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- Estimação Bayesiana
  - JAGS

```
> require(rjags)
> cat("model{
+   for (i in 1:N){
+     y[i] ~ dbern(p[i])
+     logit(p[i]) <- a[g[i]] * x[i]
+ }
+ for (j in 1:K){
+   a[j] ~ dnorm(mu.a, tau.a)
```

```
+ }
+ mu.a ~ dnorm(0, 0.0001)
+ tau.a <- pow(sigma.a, -2)
+ sigma.a ~ dunif(0, 1000)
+ }", file="av04-q3.jags")
> q3.dat <- list(y = q3.df$y, x = q3.df$x, g = q3.df$grupo, N = N, K=Ng)
> q3.model <- jags.model(file="av04-q3.jags", data=q3.dat, n.chains=3)
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 1150

Initializing model
> q3.sam <- coda.samples(q3.model, c("a", "mu.a", "sigma.a"), 20000, thin=10)
> summary(q3.sam)
Iterations = 1010:21000
Thinning interval = 10
Number of chains = 3
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

         Mean    SD Naive SE Time-series SE
a[1]    0.803 0.328  0.00423        0.00611
a[2]    0.558 0.202  0.00261        0.00304
a[3]    0.460 0.187  0.00242        0.00242
a[4]    0.539 0.200  0.00258        0.00284
a[5]    0.458 0.185  0.00239        0.00236
a[6]    0.477 0.188  0.00243        0.00243
a[7]    0.326 0.189  0.00244        0.00323
a[8]    0.601 0.218  0.00281        0.00326
a[9]    0.281 0.199  0.00257        0.00385
a[10]   0.722 0.272  0.00352        0.00507
a[11]   0.670 0.248  0.00320        0.00402
a[12]   0.561 0.203  0.00262        0.00285
a[13]   0.312 0.193  0.00249        0.00344
a[14]   0.747 0.288  0.00372        0.00496
a[15]   0.442 0.187  0.00241        0.00256
mu.a    0.530 0.115  0.00148        0.00195
sigma.a 0.261 0.147  0.00190        0.00381

2. Quantiles for each variable:

           2.5%   25%   50%   75% 97.5%
a[1]     0.3570 0.564 0.737 0.972 1.609
a[2]     0.2021 0.428 0.538 0.671 1.014
a[3]     0.0946 0.342 0.459 0.571 0.852
a[4]     0.1860 0.407 0.520 0.654 0.991
a[5]     0.1043 0.339 0.457 0.567 0.833
a[6]     0.1117 0.353 0.475 0.595 0.863
a[7]    -0.0789 0.199 0.337 0.461 0.657
a[8]     0.2453 0.455 0.572 0.719 1.108
a[9]    -0.1304 0.146 0.295 0.422 0.637
a[10]    0.3178 0.530 0.674 0.866 1.394
a[11]    0.2891 0.496 0.633 0.804 1.255
a[12]    0.2075 0.427 0.539 0.674 1.024
a[13]   -0.0884 0.185 0.325 0.451 0.657
a[14]    0.3362 0.542 0.693 0.899 1.434
a[15]    0.0613 0.326 0.444 0.558 0.820
mu.a     0.3347 0.454 0.521 0.596 0.782
sigma.a  0.0274 0.156 0.244 0.345 0.603
```

– INLA Em **INLA** é possível definir uma expressão para distribuições a priori http://www.math.ntnu.no/inla/r-inla.org/doc/prior/expression.pdf, bem como matriz de precisão para efeitos aleatórios [1]. Neste exemplo, atribuímos uma distribuiç ao $U(0, 100)$ para σ_a, tendo em conta que em **INLA** considera-se logaritmo de precisão.

```
> require(INLA)
> unif.prior = "expression:
+ a = 0;
+ b = 100;
+ sigma = sqrt(exp(-log_precision));
+ return(sigma/(b-a));"
> h.u <- list(theta=list(prior=unif.prior))
```

Neste exemplo temos um efeito aleatório multiplicando uma covariável. Para estimar um efeito aleatório desta forma, passamos a covariável como segundo argumento de `f()`.

```
> q3.inla <- inla(y ~ 0 + x + f(grupo, x, model='iid', hyper=h.u),
+                 family='binomial', data=q3.df,
+                 control.predictor=list(compute=TRUE),
+                 control.fixed=list(mean.intercept=0, prec.intercept=1/10000))
```

Resumos da distribuição marginal do parâmetro de variância dos efeitos aleatório e as médias das posterioris dos efeitos.

```
> post.sigma2 <- inla.tmarginal(function(x) (1/x), q3.inla$marginals.hy[[1]])
> inla.zmarginal(post.sigma2)
```

```
Mean            0.00369154
Stdev           0.0052907
Quantile  0.025 5.49552e-05
Quantile  0.25  0.00134229
Quantile  0.5   0.00289459
Quantile  0.75  0.00444073
Quantile  0.975 0.0110455
```

```
> q3.inla$summary.ran$grupo$mean
```

```
 [1]  0.2050302  0.0157159 -0.0634484 -0.0008029 -0.0634491 -0.0482712 -0.1758853
 [8]  0.0499123 -0.2149444  0.1429802  0.1043005  0.0157159 -0.1890371  0.1631625
[15] -0.0783353
```

4.
```
model{
  for (i in 1:N){
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- a[g[i]] * x[i] + b[g[i]]
}
for (j in 1:K){
  a[j] ~ dnorm(mu.a, tau.a)
  b[j] ~ dnorm(mu.b, tau.b)
}
mu.a ~ dnorm(0, 0.0001)
mu.b ~ dnorm(0, 0.0001)
tau <- pow(sigma, -2)
sigma ~ dunif(0, 1000)
tau.a <- pow(sigma.a, -2)
tau.b <- pow(sigma.b, -2)
sigma.a ~ dunif(0, 1000)
sigma.b ~ dunif(0, 1000)
}
```

[1] http://www.math.ntnu.no/inla/r-inla.org/doc/latent/rgeneric.pdf

```
> plot(ranef(q3.glmer)$grupo$x, q3.inla$summary.ran$grupo$mean)
> abline(0,1)
```

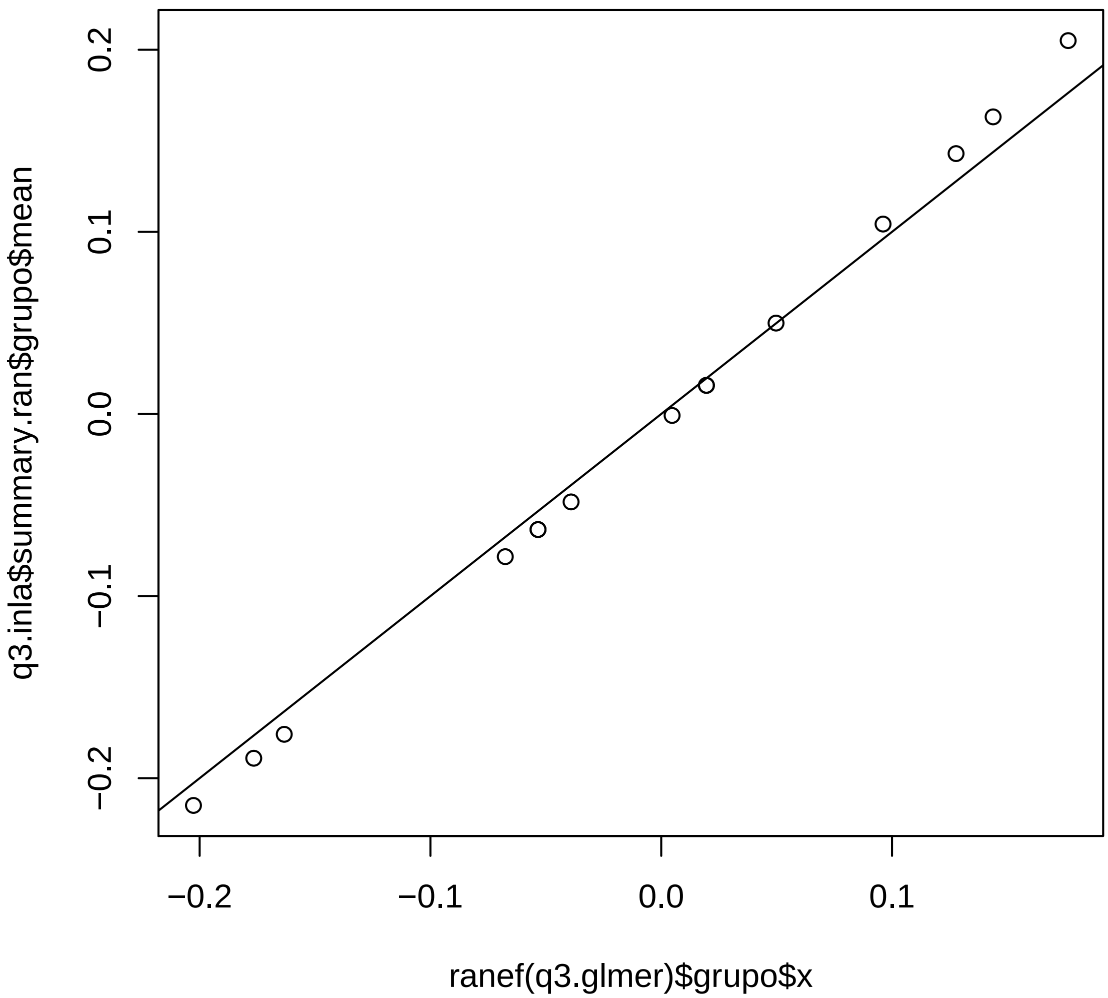


Figura 1: Comparação dos efeitos aleatórios: estimativas do `glmer()` e médias da posteriori retornada por `inla()`.

Modelo misto de regressão linear (intercepto e inclinação aleatórios):

$$
\begin{aligned}
\text{verossimilhança:} & \\
Y_i|\mu_i, \sigma^2 &\sim \mathrm{N}(\mu_i, \sigma^2) \\
\mu_i &= a_i x_i + b_i \\
\text{variáveis latentes (ef. al.):} & \\
a_j &\sim \mathrm{N}(\mu_a, \sigma_a^2) \\
b_j &\sim \mathrm{N}(\mu_b, \sigma_b^2) \\
\text{priori's:} & \\
\mu_a &\sim \mathrm{N}(0, 10000) \\
\mu_b &\sim \mathrm{N}(0, 10000) \\
\sigma_a &\sim \mathrm{U}[0, 1000] \\
\sigma_b &\sim \mathrm{U}[0, 1000]
\end{aligned}
$$

Simulação de dados do modelo proposto, com diferentes números de observações e valores da covariável para cada indivíduo.

```
> set.seed(22701)
> Nind <- 12
> (n.ind <- sample(5:12, 12, replace=TRUE))
```

```
 [1]  6  6  8  6 10 12 11  7  7  7 10  6
```

```
> q4.df <- data.frame(
+   g = rep(1:Nind, times=n.ind),
+   x = unlist(sapply(n.ind, function(n) round(sort(sample(1:20, n)))))
+   )
> ais <- rnorm(Nind, m = -1, sd = 0.2)
> bis <- rnorm(Nind, m = 50, sd = 5)
> q4.df <- transform(q4.df,
+           y = round(rnorm(nrow(q4.df), m = rep(ais, n.ind) * x + rep(bis, n.ind), sd = 3), dig=2))
```

- Estimação não-Bayesiana

  ```
  > require(lme4)
  > q4.lmer <- lmer(y ~ x + (x|g), data=q4.df, REML=FALSE)
  > summary(q4.lmer)
  ```

  ```
  Linear mixed model fit by maximum likelihood  ['lmerMod']
  Formula: y ~ x + (x | g)
     Data: q4.df

       AIC      BIC   logLik deviance df.resid
     544.2    559.6   -266.1    532.2       90

  Scaled residuals:
      Min      1Q  Median      3Q     Max
  -2.1760 -0.6353  0.0318  0.5572  2.7760

  Random effects:
   Groups   Name        Variance Std.Dev. Corr
   g        (Intercept) 52.7804  7.265
            x            0.0136  0.117    0.23
   Residual              8.8042  2.967
  Number of obs: 96, groups: g, 12

  Fixed effects:
              Estimate Std. Error t value
  (Intercept)  50.8563     2.1986    23.1
  x            -1.0014     0.0644   -15.6

  Correlation of Fixed Effects:
    (Intr)
  x -0.111
  ```

- Estimação Bayesiana
  - JAGS

```
> require(rjags)
> cat("model{
+   for (i in 1:N){
+     y[i] ~ dnorm(mu[i], tau)
+     mu[i] <- a[g[i]] * x[i] + b[g[i]]
+ }
+ for (j in 1:K){
+   a[j] ~ dnorm(mu.a, tau.a)
+   b[j] ~ dnorm(mu.b, tau.b)
+ }
+ mu.a ~ dnorm(0, 0.0001)
+ mu.b ~ dnorm(0, 0.0001)
+ tau <- pow(sigma, -2)
+ sigma ~ dunif(0, 1000)
+ tau.a <- pow(sigma.a, -2)
+ tau.b <- pow(sigma.b, -2)
+ sigma.a ~ dunif(0, 1000)
+ sigma.b ~ dunif(0, 1000)
+ }", file="av04-q4.jags")
> q4.dat <- c(q4.df, N=nrow(q4.df), K = length(unique(q4.df$g)))
> q4.model <- jags.model(file="av04-q4.jags", data=q4.dat, n.chains=3)
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 519

Initializing model
> q4.sam <- coda.samples(q4.model, c("b", "a", "sigma.b","sigma.a","sigma"), 20000, thin=10)
> summary(q4.sam)
Iterations = 1010:21000
Thinning interval = 10
Number of chains = 3
Sample size per chain = 2000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

          Mean    SD Naive SE Time-series SE
a[1]    -1.146 0.172  0.00222        0.00318
a[2]    -1.074 0.135  0.00174        0.00208
a[3]    -0.985 0.121  0.00156        0.00173
a[4]    -0.935 0.158  0.00204        0.00241
a[5]    -0.877 0.131  0.00169        0.00256
a[6]    -0.928 0.112  0.00144        0.00187
a[7]    -1.079 0.116  0.00149        0.00188
a[8]    -1.076 0.133  0.00172        0.00226
a[9]    -1.106 0.143  0.00184        0.00276
a[10]   -0.926 0.130  0.00168        0.00210
a[11]   -0.903 0.139  0.00180        0.00235
a[12]   -1.027 0.144  0.00186        0.00205
b[1]    45.687 2.083  0.02689        0.03554
b[2]    56.423 1.943  0.02508        0.02843
b[3]    35.680 1.856  0.02396        0.02577
b[4]    45.494 1.717  0.02217        0.02438
b[5]    49.650 1.620  0.02092        0.02816
b[6]    54.742 1.448  0.01870        0.02192
b[7]    44.402 1.491  0.01924        0.02304
b[8]    47.046 1.777  0.02294        0.02871
b[9]    52.000 2.211  0.02854        0.03738
b[10]   57.927 1.745  0.02253        0.02587
b[11]   60.864 1.747  0.02256        0.02824
```

```
b[12]   60.999 1.865  0.02408        0.02555
sigma    3.016 0.250  0.00322        0.00326
sigma.a  0.159 0.093  0.00120        0.00229
sigma.b  8.847 2.280  0.02944        0.02944

2. Quantiles for each variable:

            2.5%     25%    50%    75%  97.5%
a[1]    -1.53114 -1.2531 -1.121 -1.019 -0.875
a[2]    -1.37633 -1.1540 -1.060 -0.985 -0.834
a[3]    -1.22731 -1.0573 -0.988 -0.912 -0.726
a[4]    -1.22236 -1.0335 -0.948 -0.850 -0.573
a[5]    -1.09761 -0.9734 -0.891 -0.790 -0.600
a[6]    -1.12856 -1.0053 -0.936 -0.859 -0.687
a[7]    -1.32463 -1.1539 -1.070 -1.000 -0.871
a[8]    -1.36700 -1.1563 -1.064 -0.986 -0.839
a[9]    -1.42130 -1.1959 -1.089 -1.007 -0.871
a[10]   -1.15704 -1.0128 -0.939 -0.850 -0.638
a[11]   -1.14847 -0.9984 -0.917 -0.822 -0.592
a[12]   -1.33419 -1.1104 -1.020 -0.939 -0.749
b[1]    41.95854 44.2528 45.561 46.966 50.116
b[2]    52.76352 55.1420 56.346 57.627 60.510
b[3]    31.96944 34.4805 35.692 36.927 39.356
b[4]    41.97521 44.3871 45.540 46.643 48.782
b[5]    46.41211 48.5704 49.701 50.768 52.734
b[6]    51.73062 53.8032 54.791 55.734 57.371
b[7]    41.62146 43.3636 44.353 45.387 47.396
b[8]    43.67821 45.8249 47.026 48.211 50.621
b[9]    48.05765 50.4663 51.856 53.402 56.680
b[10]   54.28336 56.8204 58.001 59.105 61.238
b[11]   57.04183 59.7711 60.946 62.050 64.053
b[12]   57.43634 59.7494 60.933 62.243 64.733
sigma    2.56553  2.8404  3.002  3.170  3.553
sigma.a  0.00948  0.0908  0.151  0.215  0.362
sigma.b  5.56958  7.2797  8.452  9.985 14.504
```

– INLA

```
> require(INLA)
```

Como no exemplo anterior, precisamos definir priori Uniforme para o desvio padrão do intercepto e para a inclinação (aleatórios).

```
> unif.prior = "expression:
+ a = 0;
+ b = 100;
+ sigma = sqrt(exp(-log_precision));
+ return(sigma/(b-a));"
> h.u <- list(theta=list(prior=unif.prior))
> q4.df$g.a <- q4.df$g
> form4.inla <- y ~ x + f(g.a, model='iid', hyper=h.u) +
+     f(g, x, model='iid', hyper=h.u)
> q4.inla <- inla(form4.inla, data=q4.df)
> q4.inla$summary.fix
              mean      sd 0.025quant 0.5quant 0.975quant   mode       kld
(Intercept) 50.903 2.55149     45.801   50.906    55.9835 50.913 1.188e-10
x           -1.005 0.07382     -1.154   -1.004    -0.8602 -1.002 8.583e-11
> sigmas.post <- lapply(q4.inla$marginals.hy, function(m)
+                       inla.tmarginal(function(x) sqrt(1/x), m))
> sapply(sigmas.post, inla.zmarginal, silent=TRUE)
           Precision for the Gaussian observations Precision for g.a Precision for g
mean       2.931                                   8.385             0.1038
sd         0.2388                                  2.039             0.03319
quant0.025 2.511                                   5.235             0.04026
quant0.25  2.76                                    6.914             0.07971
quant0.5   2.911                                   8.075             0.1066
```

quant0.75	3.082	9.531	0.1249
quant0.975	3.448	13.21	0.1671