Métodos Computacionais para Inferência Estatística

Capítulo 4 - Modelos de Regressão com efeitos aleatórios

Paulo Justiniano Ribeiro Jr.
Wagner Hugo Bonat
Elias Teixeira Krainski
Walmes Marques Zeviani

LEG: Laboratório de Estatística e Geoinformação
Universidade Federal do Paraná

20^{o}SINAPE, 30-31/07/2012

Laboratório de Estatística e Geoinformação

Motivação

- Inadequação do modelo de regressão tradicional (GLM);
- Efeito estritamente linear de covariáveis contínuas;
- Observações correlacionais espaço e/ou tempo;
- Interações complexas;
- Heterogeneidade entre unidades amostrais.

Construção - Genérica

- Modelo é apresentado em uma estrutura hierárquica

  1. $[Y|\underline{b}, X] \sim f(\underline{\mu}, \phi)$
  2. $g(\underline{\mu}) = X\underline{\beta} + Z\underline{b}$
  3. $\underline{b} \sim NMV(\underline{0}, \Sigma)$.

X e Z matrizes de delineamento conhecidas.
$\underline{\beta}$, Σ e ϕ são parâmetros.

Construção - Detalhada

- Y_{ij} a j-ésima medida da u.a, $i, i = 1, \ldots, N$, $j = 1, \ldots, n_i$ e $\underline{Y}_i$ vetor de todas as medidas da u.a i. Independência condicional nos efeitos aleatórios $\underline{b}_i$, que seguem $NMV_q(\underline{0}, \Sigma)$, as respostas Y_{ij} são independentes da forma,

$$f_i(y_{ij}|\underline{b}_i, \underline{\beta}, \phi),$$

$g(\mu_{ij}) = \mathbf{x}_{ij}^T \underline{\boldsymbol{\beta}} + \mathbf{z}_{ij}^T \underline{\mathbf{b_i}}$ função de ligação $g(\cdot)$ conhecida, $\mathbf{x}_{ij}$ e $\mathbf{z}_{ij}$ covariáveis conhecidas de dimensão p e q, $\underline{\boldsymbol{\beta}}$ coeficientes de regressão fixos desconhecidos, e ϕ algum parâmetro extra na verossimilhança.

Estimação - Verossimilhança Marginal

- A contribuição para a verossimilhança da cada unidade amostral (grupo) é:

$$f_i(\underline{y}_i|\underline{\beta}, \Sigma, \phi) = \int \prod_{j=1}^{n_i} f_{ij}(y_{ij}|\underline{b}_i, \underline{\beta}, \phi) f(\underline{b}_i|\Sigma) d\underline{b}_i,$$

a verossimilhança completa para $\underline{\beta}$, Σ e ϕ é dada por

$$L(\underline{\beta}, \Sigma, \phi) = \prod_{i=1}^{N} f_i(\underline{y}_i|\underline{\beta}, \Sigma, \phi), \quad (1)$$

e sob a suposição de independência entre os grupos temos que

$$L(\underline{\beta}, \Sigma, \phi) = \prod_{i=1}^{N} \int \prod_{j=1}^{n_i} f_{ij}(y_{ij}|\underline{b}_i, \underline{\beta}, \phi) f(\underline{b}_i|\Sigma) d\underline{b}_i. \quad (2)$$

Modelo Poisson com intercepto aleatório

- Descrição do modelo
  1. $Y_{ij}|b_i \sim P(\lambda_i)$
  2. $log(\lambda_i) = \beta_0 + b_i$
  3. $b_i \sim N(0, 1/\tau^2)$
- Neste caso a contribuição para a verossimilhança de cada u.a é dada por:

$$f_i(y_{ij}|b_i, \beta_0) = \int_{-\infty}^{\infty} \prod_{j=1}^{n_i} \frac{\exp\{-\lambda_i\}\lambda_i^{y_{ij}}}{y_{ij}!} \left(\frac{\tau}{2\pi}\right)^{1/2} \exp\{-\frac{\tau^2}{2} b_i^2\} db_i$$

$$= \int_{-\infty}^{\infty} \prod_{j=1}^{n_i} \frac{\exp\{-\exp(\beta_0 + b_i)\} \exp\{(\beta_0 + b_i)^{y_{ij}}\}}{y_{ij}!}$$

$$\left(\frac{\tau}{2\pi}\right)^{1/2} \exp\{-\frac{\tau^2}{2} b_i^2\} db_i. \quad (3)$$

Modelo Poisson com intercepto aleatório

- Simular do modelo

```
> simPois <- function(f.fixo, f.aleat, beta.fixo, prec.pars, data){
+   X <- model.matrix(f.fixo, data)
+   Z <- model.matrix(f.aleat, data)
+   n.bloco <- ncol(Z)
+   n.rep <- nrow(Z)/n.bloco
+   bi <- rnorm(n.bloco,0,sd=1/prec.pars)
+   XZ <- cbind(X,Z)
+   beta <- c(beta.fixo,bi)
+   preditor <- XZ%*%beta
+   lambda <- exp(preditor)
+   y <- rpois(length(lambda),lambda=lambda)
+   return(cbind(y=y, data))
+ }
```

- Usando a função

```
> dt <- data.frame(ID = as.factor(rep(1:10, each=10)))
> set.seed(123)
> dados <- simPois(f.fixo = ~1, f.aleat= ~ -1 + ID, beta.fixo = 2, prec.pars = 3, data=dt)
> head(dados)
```

```
   y ID
1  9  1
2  7  1
3  7  1
4 13  1
5  7  1
6  7  1
```

Modelo Poisson com intercepto aleatório

- Integrando de interesse

```
> integrando <- function(b, f.fixo, beta.fixo, prec.pars,
+                        log=TRUE, dados){
+   mf <- model.frame(f.fixo, dados)
+   y <- model.response(mf)
+   X <- model.matrix(f.fixo, mf)
+   tau <- exp(prec.pars)
+   ll <- sapply(b,function(bi){
+      preditor <- X%*%beta.fixo + bi
+      lambda <- exp(preditor)
+      sum(dpois(y, lambda=lambda, log=TRUE)) +
+                  dnorm(bi, 0, sd=1/tau,log=TRUE)})
+   if(log == FALSE) ll <- exp(ll)
+   return(ll)
+ }
```

- Função escrita de forma vetorial

```
> ## Escala original
> integrando(b=c(-1,0,1), f.fixo = y~1, dados = subset(dados, ID == 1),
+            beta.fixo = 2, prec.pars=4, log=FALSE)

[1] 0.000000e+00 4.011525e-09 0.000000e+00

> ## Escala log
> integrando(b=c(-1,0,1), f.fixo = y~1, dados = subset(dados, ID == 1),
+            beta.fixo = 2, prec.pars=4, log=TRUE)

[1] -1535.10535   -19.33409 -1564.77790
```

Modelo Poisson com intercepto aleatório

- Gráfico do integrando

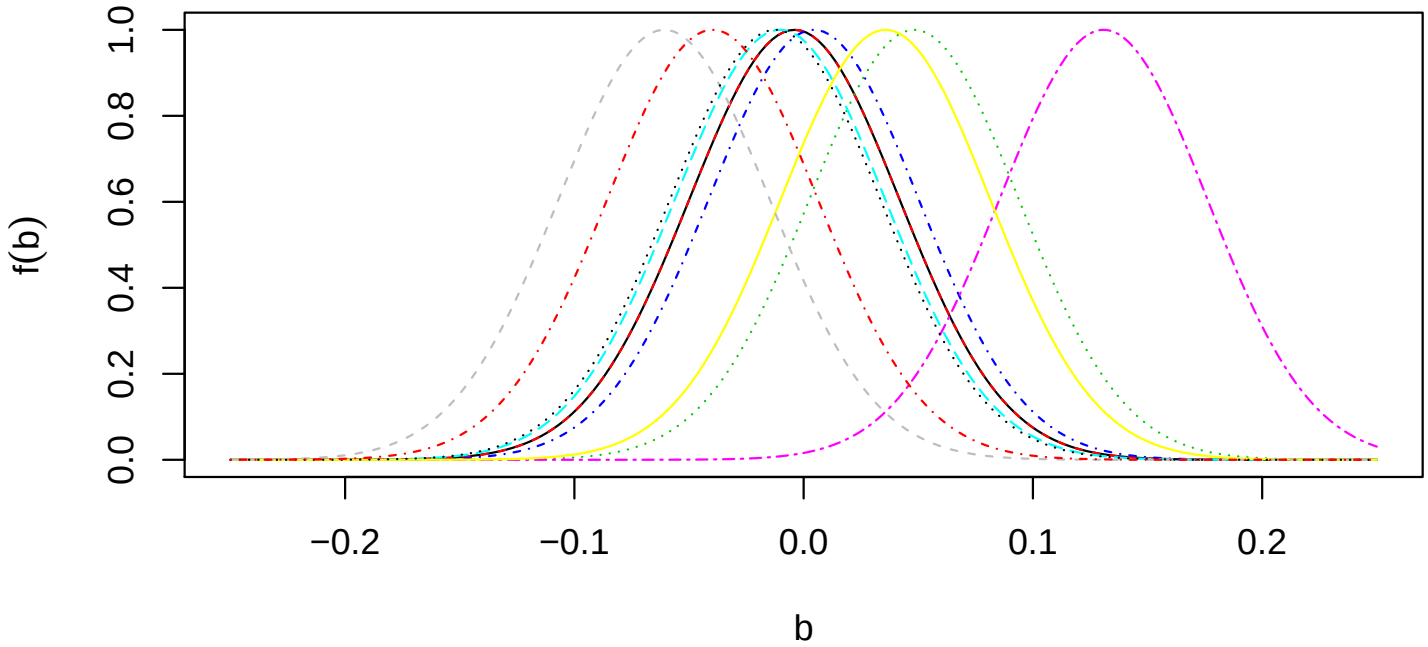


Figura 1: Integrando de acordo com unidade amostral.

Integração numérica

- Mais usados:
  1. Aproximação de Laplace;
  2. Quadratura de Gauss-Hermite;
  3. Integração Monte Carlo.
- Algumas variações:
  1. Adaptativa Gauss-Hermite;
  2. Integração Quase Monte Carlo;
  3. Adaptativa Monte Carlo e Quase Monte Carlo.
- Métodos básicos (não muito utéis em GLMM)
  1. Método Trapezoidal;
  2. Método de Simpson;
  3. Outros.

Aproximação de Laplace

- A integral da função $f(x)$ é aproximada por:

$$\int \exp\{Q(x)\}dx \approx (2\pi)^{n/2}|Q''(\hat{x})|^{-1/2}\exp\{Q(\hat{x})\}$$

$Q(x) = \log(f(x))$ e $\hat{x}$ é o ponto de máximo de $Q(x)$.

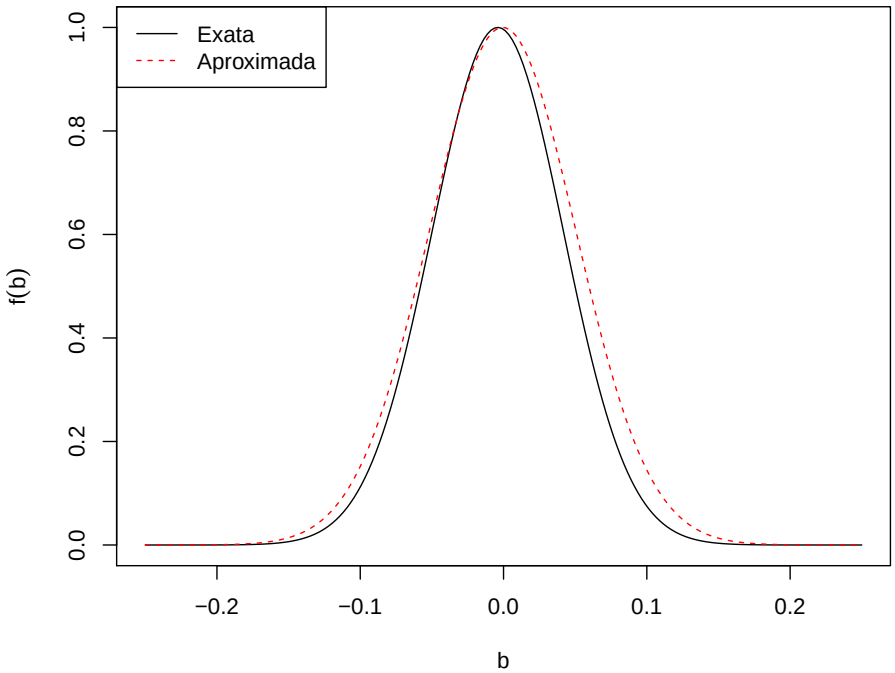
- Em R pode-se programar genericamente este método,

```
> laplace <- function(funcao, otimizador,n.dim, ...){
+   integral <- -999999
+   inicial <- rep(0,n.dim)
+   temp <<- try(optim(inicial, funcao,..., method=otimizador,
+                hessian=TRUE, control=list(fnscale=-1)))
+   if(class(temp) != "try-error"){
+   integral <- exp(temp$value) * (exp(0.5*log(2*pi) -
+                   0.5*determinant(-temp$hessian)$modulus))}
+   return(integral)
+ }
```

Aproximação de Laplace

- Exemplo - Modelo Poisson com intercepto aleatório

```
> ## Para a primeira u.a
> log(laplace(integrando, otimizador="BFGS", n.dim=1,
+           f.fixo = y~1, dados=subset(dados, ID == 1),
+           beta.fixo = 2, prec.pars=4, log=TRUE))

[1] -22.42681
attr(,"logarithm")
[1] TRUE
```



Quadratura de Gauss-Hermite

- Resolve integrais da forma:

$$\int_{-\infty}^{\infty} e^{-x^2} f(x) dx$$

- A integral é aproximada por uma soma ponderada,

$$\int_{-\infty}^{\infty} e^{-x^2} f(x) dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

onde n é o número de pontos usadas para a aproximação. Os x_i são as raízes do polinômio de Hermite $H_n(x)(i = 1 < 2, \ldots, n)$ e os pesos w_i associados são dados por

$$w_i = \frac{2^{n-1} n! \sqrt{\pi}}{n^2 [H_{n-1}(x_i)]^2}$$

Quadratura de Gauss-Hermite

- Em R este método pode facilmente ser implementado,

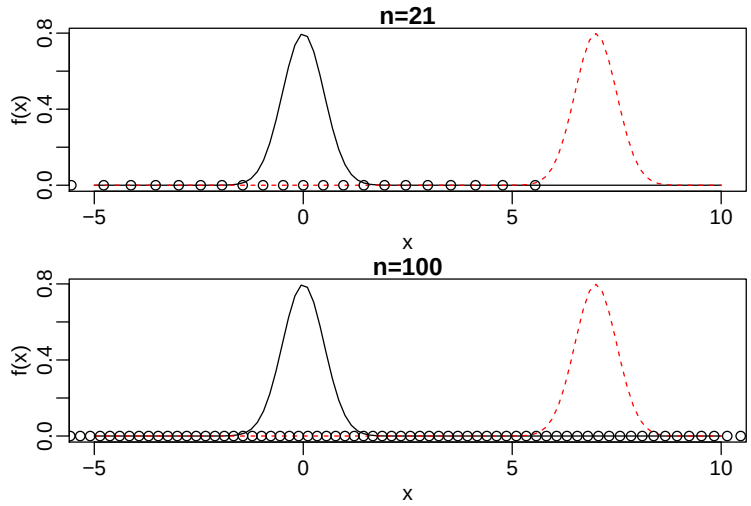
```
> require(statmod)
> gauss.hermite <- function(integrando, n.pontos, ...){
+   pontos <- gauss.quad(n.pontos, kind="hermite")
+   integral <- sum(pontos$weights*integrando(pontos$nodes,...)/exp(-pontos$nodes^2))
+   return(integral)
+ }
```

- Usando a função,

```
> log(gauss.hermite(integrando = integrando, n.pontos=21,
+                   f.fixo = y~1, dados=subset(dados, ID == 1),
+                   beta.fixo = 2, prec.pars=4, log=FALSE))
```

```
[1] -20.0701
```

- Graficamente



Integração Monte Carlo

- Resolve integrais da forma,

$$I = \int_D f(x)dx \tag{4}$$

- Seja uma função densidade de probabilidade $p(x)$ cujo domínio coincide com D, então

$$I = \int_D \frac{f(x)}{p(x)} p(x)dx.$$

- $E\left(\frac{f(x)}{p(x)}\right)$ que pode ser estimado gerando número aleatórios de acordo com $p(x)$.
- Avalia-se $f(x)/p(x)$ para cada amostra e calcula-se a média.
- Quanto mais amostras foram geradas esta média converge para o verdadeiro valor da integral.

Integração Monte Carlo

- Em R implementamos este método,

```
> require(fOptions)
> require(mvtnorm)
> monte.carlo <- function(funcao, n.dim, n.pontos, tipo, ...){
+ if(tipo == "MC"){ pontos <- rmvnorm(n.pontos,mean=rep(0,n.dim))}
+ if(tipo == "Halton"){ pontos <- rnorm.halton(n.pontos,n.dim)}
+ if(tipo == "Sobol"){ pontos <- rnorm.sobol(n.pontos,n.dim)}
+ norma <- apply(pontos,1,dmvnorm)
+ integral <- mean(funcao(pontos,...)/norma)
+ return(integral)
+ }
```

- Aplicando para calcular a integral no Modelo de Poisson,

```
> log(monte.carlo(integrando, n.dim=1, tipo = "MC", n.pontos=20,
+          f.fixo = y~1, dados=subset(dados, ID == 1),
+          beta.fixo = 2, prec.pars=4, log=FALSE))

[1] -21.47252

> log(monte.carlo(integrando, n.dim=1, tipo = "Halton", n.pontos=20,
+          f.fixo = y~1, dados=subset(dados, ID == 1),
+          beta.fixo = 2, prec.pars=4, log=FALSE))

[1] -21.41082

> log(monte.carlo(integrando, n.dim=1, tipo = "Sobol", n.pontos=20,
+          f.fixo = y~1, dados=subset(dados, ID == 1),
+          beta.fixo = 2, prec.pars=4, log=FALSE))

[1] -21.41079
```

Laboratório de Estatística e Geoinformação

Integração Monte Carlo

- Graficamente podemos visualizar onde os pontos são colocados

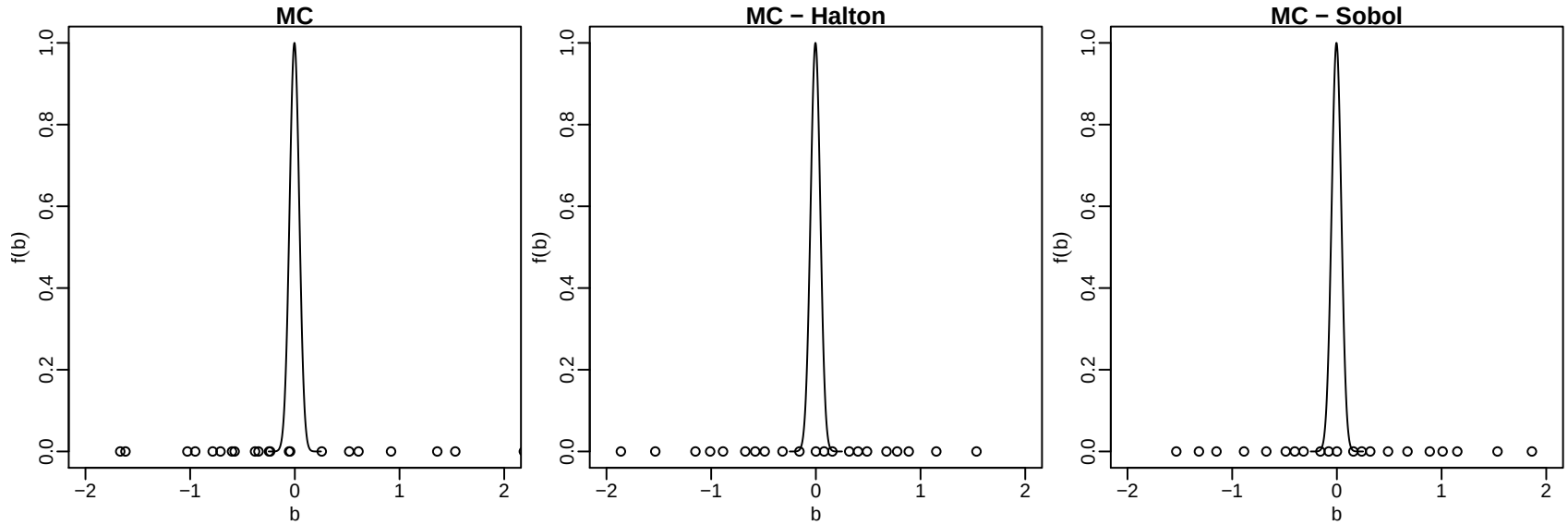


Figura 4: Espalhamento dos pontos de integração pelo método de Monte Carlo.

Comparando os métodos

```
> ## Laplace
> system.time(print(log(laplace(integrando, otimizador="BFGS", n.dim=1,
+          f.fixo = y~1, dados=subset(dados, ID == 1),
+          beta.fixo = 2, prec.pars=4, log=TRUE))))

[1] -22.42681
attr(,"logarithm")
[1] TRUE
   user  system elapsed
  0.064   0.000   0.064

> ## GH
> system.time(print(log(gauss.hermite(integrando = integrando, n.pontos=21,
+                   f.fixo = y~1, dados=subset(dados, ID == 1),
+                   beta.fixo = 2, prec.pars=4, log=FALSE))))

[1] -20.0701
   user  system elapsed
  0.008   0.004   0.006

> ## MC
> system.time(print(log(monte.carlo(integrando, n.dim=1, tipo = "Halton", n.pontos=20,
+          f.fixo = y~1, dados=subset(dados, ID == 1),
+          beta.fixo = 2, prec.pars=4, log=FALSE))))

[1] -21.41082
   user  system elapsed
  0.024   0.024   0.023

>
```

Verossimilhança Marginal

- Genericamente podemos escrever a função de verossimilhança marginal

```
> veroM <- function(modelo, formu.X, formu.Z, beta.fixo, prec.pars,
+                   integral, pontos, otimizador, n.dim, dados){
+   dados.id <- split(dados, dados$ID)
+   ll <- c()
+   for(i in 1:length(dados.id)){
+     X <- model.matrix(as.formula(formu.X),data=dados.id[[i]])
+     Z <- model.matrix(as.formula(formu.Z),data=dados.id[[i]])
+     if(integral == "LAPLACE"){
+     ll[i] <- laplace(modelo,otimizador=otimizador,n.dim=n.dim,
+                X=X, Z=Z, Y=dados.id[[i]]$y, beta.fixo=beta.fixo,
+                prec.pars=prec.pars,log=TRUE)}
+     if(integral == "GH"){
+     ll[i] <- gauss.hermite.multi(modelo, n.pontos= pontos, n.dim=n.dim,
+                     X=X, Z=Z, Y=dados.id[[i]]$y, beta.fixo=beta.fixo,
+                     prec.pars=prec.pars,log=FALSE)}
+     if(integral == "MC"){
+     ll[i] <- monte.carlo(modelo,n.pontos=pontos, n.dim=n.dim,
+                     tipo= "MC", X=X, Z=Z, Y=dados.id[[i]]$y,
+                     beta.fixo=beta.fixo, prec.pars=prec.pars,
+                     log=FALSE)}
+     if(integral == "QMH"){
+     ll[i] <- monte.carlo(modelo,n.pontos=pontos, n.dim=n.dim,
+                     tipo= "Halton", X=X, Z=Z, Y=dados.id[[i]]$y,
+                     beta.fixo=beta.fixo,
+                     prec.pars=prec.pars,log=FALSE)}
+     if(integral == "QMS"){
+     ll[i] <- monte.carlo(modelo,n.pontos=pontos, n.dim=n.dim,
+                    tipo= "Sobol", X=X, Z=Z, Y=dados.id[[i]]$y,
+                    beta.fixo=beta.fixo, prec.pars=prec.pars,log=F)}
+     }
+   return(sum(log(ll)))
+ }
```

Estimação - Poisson com intercepto aleatório

- Integrando de cada unidade amostral

```
> Poisson.Int <- function(b,beta.fixo, prec.pars, X, Z, Y,log=TRUE){
+   tau <- exp(prec.pars)
+   ll = sapply(b,function(bi){
+     preditor <- as.matrix(X)%*%beta.fixo + as.matrix(Z)%*%bi
+     lambda <- exp(preditor)
+     sum(dpois(Y,lambda=lambda,log=TRUE)) +
+         dnorm(bi, 0, sd = 1/tau , log=TRUE)
+   })
+   if(log == FALSE){ll <- exp(ll)}
+   return(ll)}
```

- Verossimilhança marginal completa

```
> mod.Poisson <- function(b0,tau,integral, pontos, otimizador, n.dim, dados){
+   ll = veroM(modelo = Poisson.Int, formu.X="~1", formu.Z="~1",
+              beta.fixo = b0, prec.pars=tau, integral=integral,
+              pontos=pontos,otim=otimizador,n.dim=n.dim,dados=dados)
+   #print(round(c(b0,tau,ll),2))
+   return(-ll)}
```

Maximizando a Verossimilhança Marginal

- Maximizando a verossimilhança marginal

```
> require(bbmle)
> system.time(P.laplace <- mle2(mod.Poisson,start=list(b0=0,tau=log(1/4)),
+   data=list(integral="LAPLACE",otimizador = "BFGS", n.dim=1,
+   dados=dados, pontos=NA)))

   user  system elapsed
 16.037   0.220  16.046
```

- Resumo do ajuste

```
> summary(P.laplace)

Maximum likelihood estimation

Call:
mle2(minuslogl = mod.Poisson, start = list(b0 = 0, tau = log(1/4)),
    data = list(integral = "LAPLACE", otimizador = "BFGS", n.dim = 1,
        dados = dados, pontos = NA))

Coefficients:
    Estimate Std. Error z value     Pr(z)
b0  2.019228   0.093016 21.7083 < 2.2e-16 ***
tau 1.311171   0.262178  5.0011 5.701e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

-2 log L: 500.6696
```

Laboratório de Estatística e Geoinformação

Poisson com intercepto aleatório - Versão Bayesiana

- Especificando o modelo em JAGS

```
> mod.poisson <- function(){
+   for(j in 1:n.ua){
+         for(i in 1:n.rep){
+                 Y[i,j] ~ dpois(lambda[i,j])
+                 log(lambda[i,j]) <- beta0 + b[j]
+         }
+         b[j] ~ dnorm(0, tau)
+   }
+   beta0 ~ dnorm(0, 0.001)
+   tau ~ dgamma(1, 1)
+ }
```

Código - Versão Bayesiana

- Chamando o JAGS do R para rodar o MCMC

```
> require(dclone)
> y.mat <- matrix(dados$y, 10, 10)
> dados.list <- list(Y = y.mat, n.ua = 10, n.rep = 10)
> bayesPois <- jags.fit(dados.list,c("beta0","tau"),mod.poisson,
+                       n.adapt = 1000, n.update = 1000, thin = 5,
+                       n.iter=10000, n.chains=5)

> summary(bayesPois)

Iterations = 2005:12000
Thinning interval = 5
Number of chains = 5
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
beta0 2.002 0.1719 0.001719       0.007646
tau   3.845 1.6725 0.016725       0.020893

2. Quantiles for each variable:

       2.5%   25%   50%   75% 97.5%
beta0 1.650 1.896 2.001 2.110 2.345
tau   1.316 2.647 3.603 4.785 7.805
```

Modelo Beta Longitudinal

- Modelo Beta Longitudinal

$$Y_{it}|b_{ji} \sim B(\mu_{it}, \phi)$$
$$g(\mu_{it}) = (\beta_0 + b_{0i}) + (\beta_1 + b_{1i})t$$
$$\begin{bmatrix} b_{0i} \\ b_{1i} \end{bmatrix} \sim NM_2\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_I^2 & \rho \\ \rho & \sigma_S^2 \end{bmatrix}\right)$$

- Função para simular deste modelo

```
> inv.logit <- function(x){exp(x)/(1+exp(x))}
> rbeta.model <- function(ID, tempo, beta.fixo, prec.pars){
+   dados = data.frame("ID" = rep(1:ID,each=tempo),
+           "cov" = rep(seq(0, 1,l=tempo),ID))
+   dados.id <- split(dados,dados$ID)
+   cov.rho <- prec.pars[3]*sqrt(prec.pars[1])*sqrt(prec.pars[2])
+   Sigma<-matrix(c(prec.pars[1],cov.rho,cov.rho,prec.pars[2]),2,2)
+   y <- matrix(NA, ncol=ID, nrow=tempo)
+   for(i in 1:ID){
+     X <- model.matrix(~cov, data=dados.id[[i]])
+     Z <- model.matrix(~cov, data=dados.id[[i]])
+     b <- rmvnorm(n=1,mean=c(0,0),sigma=Sigma)
+     preditor <- X%*%as.numeric(beta.fixo) + Z%*%as.numeric(b)
+     mu <- inv.logit(preditor)
+     y[,i]<-rbeta(length(mu),mu*prec.pars[4],
+                  (1-mu)*prec.pars[4])
+   }
+   dados$y <- c(y)
+   return(dados)
+ }
```

Modelo Beta Longitudinal

- Conjunto de dados simulado

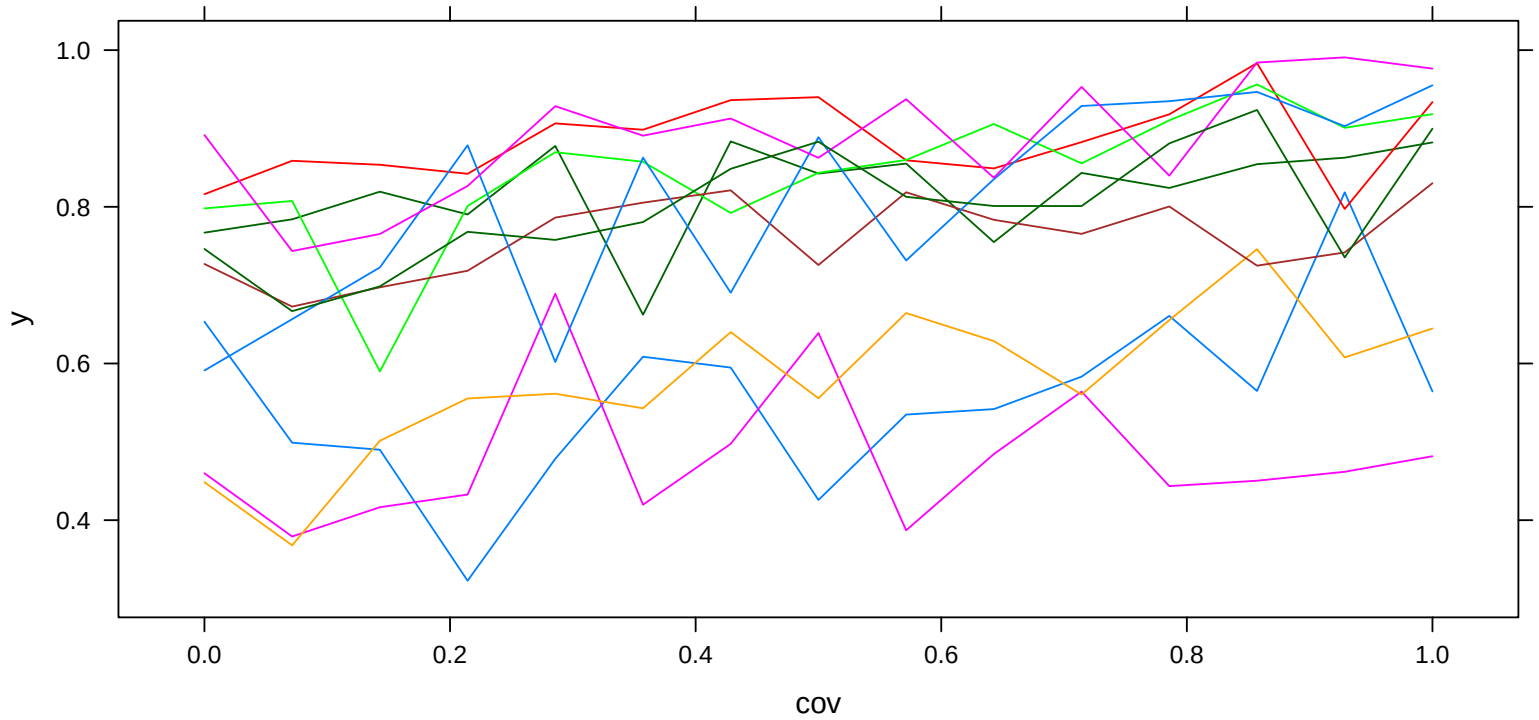


Figura 5: Trajetórias por unidade amostral - Modelo Beta longitudinal.

Modelo Beta Longitudinal

- Escrevendo a integral de interesse,

```
> transf.rho <- function(rho){
+   -1+2*(exp(rho)/(exp(rho)+1))
+ }
> vero.slope <- function(uv,beta.fixo, prec.pars, X, Z, Y,log=TRUE){
+   sigmaI <- exp(prec.pars[1])^2
+   sigmaS <- exp(prec.pars[2])^2
+   rho <- transf.rho(prec.pars[3])
+   phi <- exp(prec.pars[4])
+   cov.rho <- rho*(sqrt(sigmaI)*sqrt(sigmaS))
+   if(class(dim(uv)) == "NULL"){uv <- matrix(uv,1,2)}
+   ll = apply(uv,1,function(uvi){
+     preditor <- X%*%beta.fixo + Z%*%as.numeric(uvi)
+     mu <- inv.logit(preditor)
+     sigma <- matrix(c(sigmaI,cov.rho,cov.rho,sigmaS),2,2)
+     sum(dbeta(Y, mu*phi, (1-mu)*phi, log=TRUE)) +
+         dmvnorm(uvi, c(0,0), sigma = sigma , log=TRUE)})
+   if(log == FALSE){ll <- exp(ll)}
+   return(ll)}
```

Laboratório de Estatística e Geoinformação

Modelo Beta Longitudinal

- Escrevendo a verossimilhança marginal

```
> model.Beta <- function(b0, b1, sigmaI, sigmaS, rho, phi, otimizador,
+     n.dim, dados) {
+     ll = veroM(modelo = vero.slope, formu.X = "~cov", formu.Z = "~cov",
+         beta.fixo = c(b0, b1), prec.pars = c(sigmaI, sigmaS,
+             rho, phi), integral = integral, pontos = pontos,
+         otimizador = otimizador, n.dim = n.dim, dados = dados)
+     return(-ll)
+ }
```

- Maximizando a verossimilhança marginal

```
> ajuste = mle2(model.Beta, start=list(b0=0, b1=0,sigmaI=-0.5,
+              sigmaS=-0.5, rho = 0.3, phi = log(25)), method="BFGS",
+              data=list(integral="LAPLACE", pontos=1, otimizador="BFGS",
+              n.dim=2,dados=dados))
```

Laboratório de Estatística e Geoinformação

Modelo Beta Longitudinal

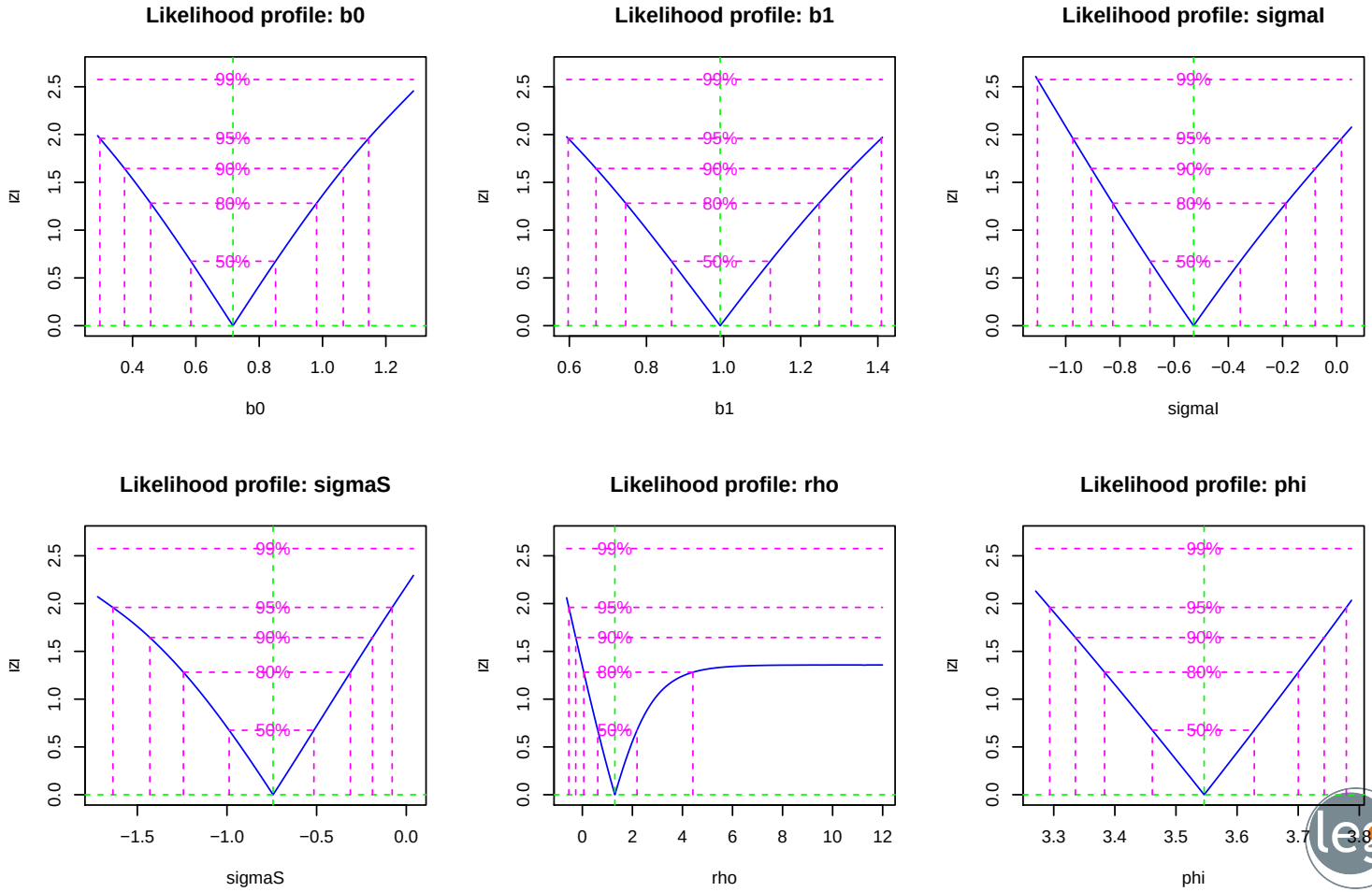
- Resumo do modelo

```
> summary(ajuste)

Maximum likelihood estimation

Call:
mle2(minuslogl = model.Beta, start = list(b0 = 0, b1 = 0, sigmaI = -0.5,
    sigmaS = -0.5, rho = 0.3, phi = log(25)), method = "BFGS",
    data = list(integral = "LAPLACE", pontos = 1, otimizador = "BFGS",
        n.dim = 2, dados = dados))

Coefficients:
       Estimate Std. Error z value     Pr(z)
b0      0.71712    0.36165  1.9829   0.04738 *
b1      0.99116    0.02887 34.3321 < 2.2e-16 ***
sigmaI -0.52810    0.73933 -0.7143   0.47504
sigmaS -0.74294    0.49673 -1.4957   0.13474
rho     1.29939    0.27167  4.7829 1.727e-06 ***
phi     3.54610    0.08744 40.5547 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

-2 log L: -334.7962
```

Modelo Beta Longitudinal

- Perfil de verossimilhança

```
> perfil <- profile(ajuste)
```



Modelo Beta Longitudinal - Versão Bayesiana

- Escrevemos o modelo em JAGS

```
> beta.longitudinal <- function(){
+   for(j in 1:n.ua){
+         for(i in 1:n.rep){
+       Y[j,i] ~ dbeta(mu[j,i]*phi,(1-mu[j,i])*phi)
+       logit(mu[j,i]) <- X.int[j,i]*beta0 + X.slope[j,i]*beta1 +
+                         Z.int[j,i]*b0[j] + Z.slope[j,i]*b1[j]
+     }
+     b0[j] <- inter.slope[j,1]
+     b1[j] <- inter.slope[j,2]
+     inter.slope[j,1:2] ~ dmnorm(mu.theta[j,1:2], Pr.theta[1:2,1:2])
+     mu.theta[j,1] <- 0
+     mu.theta[j,2] <- 0
+   }
+   Sigma[1,1] <- tau.I^2
+   Sigma[2,2] <- tau.S^2
+   Sigma[1,2] <- rho*(tau.I*tau.S)
+   Sigma[2,1] <- rho*(tau.I*tau.S)
+   Pr.theta <- inverse(Sigma)
+   beta0 ~ dnorm(0, 0.01)
+   beta1 ~ dnorm(0, 0.01)
+   phi   ~ dgamma(1, 0.01)
+   tau.I ~ dgamma(1,0.01)
+   tau.S ~ dgamma(1,0.01)
+   rho   ~ dunif(-1,1)
+ }
```

Laboratório de Estatística e Geoinformação

Modelo Beta Longitudinal - Versão Bayesiana

- Arrumando os dados para passar pro JAGS

```
> ## Calculando o numero de repeticoes e de individuos
> n.ua <- length(unique(dados$ID))
> n.rep <- length(unique(dados$cov))
> ## Arrumando os dados para o JAGS
> y.mat <- t(matrix(dados$y, n.rep, n.ua))
> ## Separando os dados por id
> dados.id <- split(dados,dados$ID)
> ## Matriz da parte fixa
> X.int <- model.matrix(~1, data=dados.id[[1]])
> X.slope <- model.matrix(~cov-1, data=dados.id[[1]])
> X.int <- matrix(rep(X.int,n.ua),n.rep,n.ua)
> X.slope <- matrix(rep(X.slope,n.ua),n.rep,n.ua)
> ## Matriz da parte aleatoria
> Z.int <- X.int
> Z.slope <- X.slope
> ## Montando a lista para o JAGS
> dat.jags <- list(Y = y.mat, X.int = t(X.int), X.slope = t(X.slope),
+   Z.int = t(Z.int), Z.slope = t(Z.slope), n.ua = n.ua, n.rep = n.rep)
```

Modelo Beta Longitudinal - Versão Bayesiana

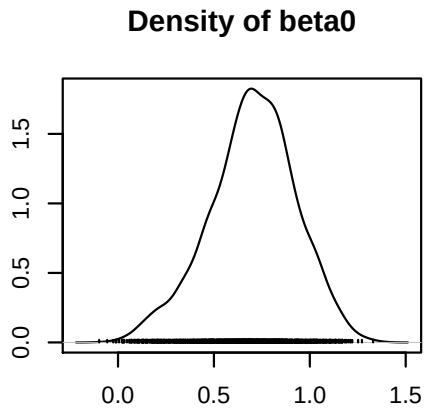
- Ajustando o modelo via JAGS

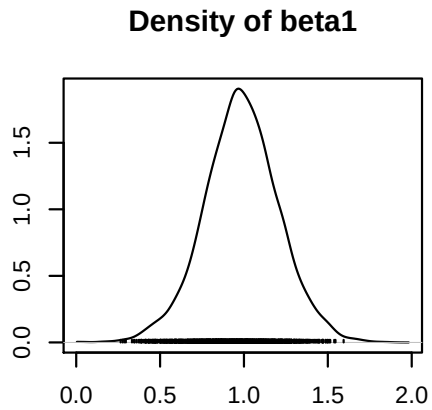
```
> beta.model <- jags.fit(dat.jags,
+                  c("beta0","beta1", "tau.I", "tau.S", "phi", "rho"),
+                  beta.longitudinal, n.adapt = 2000, n.update = 1000,
+                  thin = 5, n.iter= 10000, n.chains=3)

> load("beta.model.RData")

> summary(beta.model)

Iterations = 3005:13000
Thinning interval = 5
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
beta0  0.7006 0.2271 0.002932       0.016638
beta1  0.9845 0.2232 0.002881       0.013566
phi   34.9609 4.2382 0.054714       0.068411
rho    0.3887 0.3378 0.004361       0.011265
tau.I  0.7633 0.2381 0.003074       0.005945
tau.S  0.6337 0.2498 0.003224       0.008428

2. Quantiles for each variable:

         2.5%     25%     50%     75%   97.5%
beta0  0.2046  0.5614  0.7092  0.8519  1.1249
beta1  0.5270  0.8425  0.9826  1.1273  1.4299
phi   27.1820 31.9867 34.8490 37.6398 43.8022
rho   -0.3589  0.1606  0.4316  0.6514  0.9035
tau.I  0.4333  0.6025  0.7215  0.8762  1.3309
tau.S  0.2735  0.4641  0.5936  0.7578  1.2121
```

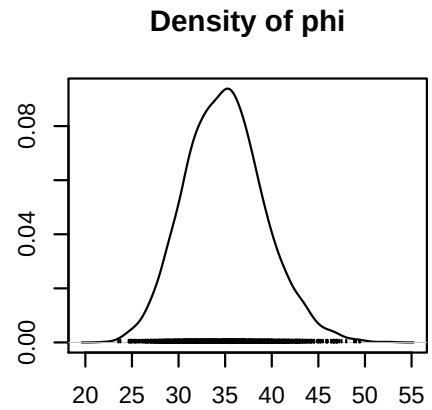
Laboratório de Estatística e Geoinformação

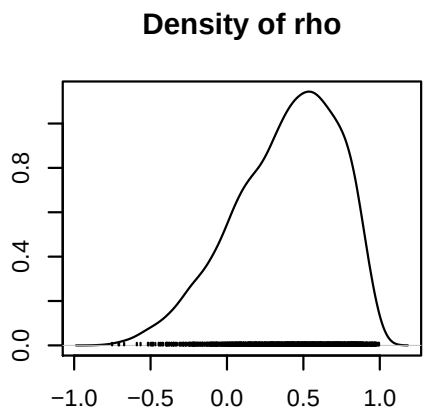
Modelo Beta Longitudinal - Versão Bayesiana

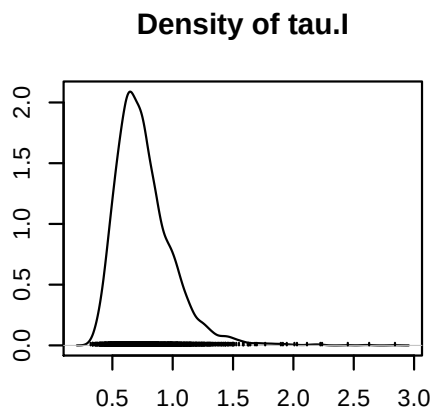
- Distribuições a posteriori

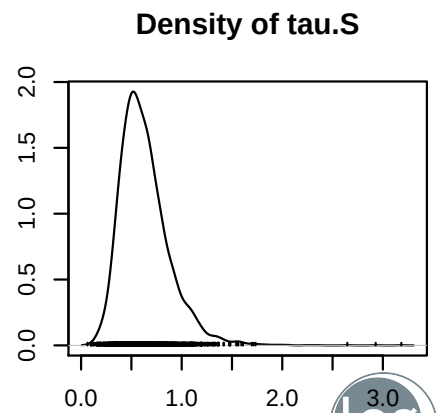












Laboratório de Estatística e Geoinformação

Modelo de Teoria de Resposta ao Item

- Modelo de três parâmetros

$$P(Y_{ij}|\theta_j) = (1 - c_i) + \frac{c_i}{1 + \exp\{-\alpha_i(\theta_j - \beta_i)\}}.$$

- Modelo completo

$$Y_{ij}|\theta_j \sim B(n = 1, p_{ij})$$
$$\theta_j \sim N(0, 1).$$

Laboratório de Estatística e Geoinformação

TRI - Implementação

- Modelo ML3

```
> logistico <- function(beta, alpha, ce, theta){
+         return(ce + (1-ce)* (1/(1+ exp(-alpha*(theta-beta)))))
+ }
```

- Simular do modelo

```
> simula.tri <- function(n.ind, beta, alpha, ce){
+   theta <- rnorm(n.ind, 0, 1)
+   p <- matrix(NA, ncol = length(beta), nrow = n.ind)
+   y <- p
+   for(i in 1:length(beta)){
+   p[,i] <- logistico(beta = beta[i], alpha = alpha[i],
+                      ce = ce[i], theta=theta)}
+   for(i in 1:n.ind){
+   y[i,] <- rbinom(n=length(beta), size = 1, p = p[i,])}
+   dados <- data.frame(y = y, ID = 1:100)
+   return(dados)}

> set.seed(123)
> dados <- simula.tri(n.ind=100, beta=c(-2,-1, 0, 1, 2), alpha=c(1,1,1,1,1),
+                     ce=c(0,0,0,0,0))
> head(dados)

  y.1 y.2 y.3 y.4 y.5 ID
1   1   0   0   0   0  1
2   0   1   0   0   0  2
3   1   1   1   0   0  3
4   1   1   1   1   1  4
5   1   0   0   0   0  5
6   1   1   1   0   1  6
```

TRI - Implementação

- Integrando modelo de Rasch

```
> integrando <- function(theta, b1, b2,b3, b4, b5, y, log = FALSE){
+   beta <- c(b1,b2,b3,b4,b5)
+   n.beta <- length(beta)
+   p <- matrix(NA, ncol = n.beta, nrow = 1)
+   ll = sapply(theta, function(thetai){
+     for(i in 1:n.beta){
+       p[,i] <- logistico(beta = beta[i], alpha = 1, ce = 0,
+                          theta=thetai)
+     }
+     sum(dbinom(y, size = 1, prob = p, log=TRUE)) +
+           dnorm(thetai, 0, 1, log=TRUE)})
+   if(log == FALSE){ll <- exp(ll)}
+   return(ll)
+ }
```

- Verossimilhança marginal

```
> marginal <- function(b1, b2, b3, b4, b5, dados){
+   y.id <- split(dados, dados$ID)
+   beta <- c(b1,b2,b3,b4,b5)
+   print(round(beta,4))
+   n.beta <- length(beta)
+   integral <- c()
+   for(i in 1:length(y.id)){
+   integral[i] <- integrate(integrando, lower= -Inf, upper = Inf,
+                     b1 = b1, b2 = b2, b3 = b3, b4 = b4, b5 = b5,
+                     y = as.numeric(y.id[[i]][1:n.beta]))$value}
+   ll <- sum(log(integral))
+   return(-ll)
+ }
```

TRI - Implementação

- Maximizando a verossimilhança marginal

```
> ajuste <- mle2(marginal, start=list(b1=0,b2=0,b3=0,b4=0,b5=0),
+                data=list(dados=dados))
```

- Resumo do modelo

```
> summary(ajuste)

Maximum likelihood estimation
Coefficients:
    Estimate Std. Error z value     Pr(z)
b1 -1.636976   0.286718 -5.7093 1.134e-08 ***
b2 -0.641374   0.247421 -2.5922  0.009535 **
b3  0.090727   0.241169  0.3762  0.706770
b4  1.063558   0.260293  4.0860 4.389e-05 ***
b5  1.864777   0.303266  6.1490 7.799e-10 ***

-2 log L: 575.0457
```

- Usando o ltm

```
> require(ltm)

> rasch(dados, constraint=cbind(length(dados)+1, 1))

Coefficients:
Dffclt.y.1  Dffclt.y.2  Dffclt.y.3  Dffclt.y.4  Dffclt.y.5       Dscrmn
    -1.637      -0.641       0.091       1.064       1.865       1.000

Log.Lik: -287.523
```