Implementação em R de modelos de regressão binária com ligação paramétrica

Bernardo Pereira dos Santos

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa: Estatística

Orientador: Prof. Dr. Carlos Alberto de Bragança Pereira

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro do CNPq

São Paulo, março de 2013

Implementação em R de modelos de regressão binária com ligação paramétrica

Esta versão da dissertação contém as correções e alterações sugeridas pela Comissão Julgadora durante a defesa da versão original do trabalho realizada em 27/02/2013. Uma cópia da versão original está disponível no Instituto de Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

- Prof. Dr. Carlos Alberto de Bragança Pereira (orientador) – IME-USP
- Prof. Dr. Adriano Polpo de Campos – Ufscar
- Prof. Dr. Enrico Antônio Colosino – UFMG

Agradecimentos

Gostaria de agradecer a meus pais Sylvio José Pereira dos Santos e Maria Cristina dos Santos pelo apoio e por não me deixarem desistir.

Um agradecimento especial para minha querida Patricia Schlithler da Fonseca Cardoso pelo carinho, paciência e revisão de texto.

Não posso deixar de agradecer meu orientador Prof. Dr. Carlos Alberto de Bragança Pereira e ao Prof. Dr. Adriano Polpo de Campos pelas contribuições valiosas para o trabalho.

Por fim, agradeço ao CNPq pelo auxílio financeiro durante o período do mestrado.

Resumo

dos Santos, B. P. **Implementação em R de modelos de regressão binária com ligação paramétrica**. 2013. 75 pp. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2013.

A análise de dados binários é usualmente feita através da regressão logística, mas esse modelo possui limitações. Modificar a função de ligação da regressão permite maior flexibilidade na modelagem e diversas propostas já foram feitas nessa área. No entanto, não se sabe de nenhum pacote estatístico capaz de estimar esses modelos, o que dificulta sua utilização. O presente trabalho propõe uma implementação em R de quatro modelos de regressão binária com função de ligação paramétrica usando tanto a abordagem frequentista como a Bayesiana.

Palavras-chave: dados binários, modelos lineares generalizados, função de ligação.

Abstract

dos Santos, B. P. **R implementation of binary regression models with parametric link**. 2013. 75 pp. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2013.

Binary data analysis is usually conducted with logistic regression, but this model has limitations. Modifying the link function allows greater flexibility in modelling and several proposals have been made on the field. However, to date there are no packages capable of estimating these models imposing some difficulties to utilize them. The present work develops an R implementation of four binary regression models with parametric link functions in both frequentist and Bayesian approaches.

Keywords: binary data, generalized linear models, link function.

Sumário

Capítulo 1

Introdução

Fenômenos que assumem apenas dois estados são chamados de binários e seu estudo sempre foi de grande interesse. Seja prever o resultado de uma eleição ou estudar a prevalência do uso de drogas ou remédios em uma certa população, é comum a presença de dados dicotômicos e o problema de estimar a proporção de ocorrência de um dos eventos faz-se importante. Além disso, pode ser interessante avaliar quanto algumas variáveis influenciam na proporção desse evento. No entanto, técnicas de regressão linear não podem ser utilizadas devido a violação de algumas suposições (Kutner *et al.*, 2005).

Diversas soluções para esse problema foram propostas e a regressão logística, um caso particular dos modelos lineares generalizados (Nelder e Wedderburn, 1972), é uma das mais utilizadas. Nesses, uma função de ligação é aplicada na combinação linear dos preditores de modo que a resposta seja um valor apropriado para o problema. Para dados binários, é natural utilizar funções quantil* como é o caso da regressão logística que utiliza a distribuição logística como o nome sugere. Outras distribuições como a normal padrão (probito) e valor extremo (complementar log-log) também são opções comuns para a função de ligação (Agresti, 2002).

Diversas outras funções de ligações foram propostas na literatura para estender as mencionadas. Prentice (1975) apresentou uma função de ligação baseada no logaritmo da distribuição F-Snedecor. Aranda-Ordaz (1981) propôs uma função assimétrica que tem os modelos logístico e complementar log-log como casos particulares. Stukel (1988) introdu-

*Inversa da função de distribuição.

ziu uma generalização do modelo logístico adicionando dois parâmetros à função de ligação logística que controlam o comportamento das caudas da distribuição. Caron *et al.* (2009) e Caron (2010) sugerem um modelo baseado na distribuição Weibull que possui o modelo complementar log-log como caso especial e aproxima os modelos logístico e probito.

Apesar das qualidades destes modelos, até onde se sabe, não existe função para a estimação desses nos pacotes estatísticos a não ser pelos três primeiros casos mencionados. Todas as outras funções compartilham a qualidade de serem paramétricas, ou seja, há elementos extras a serem estimados além dos coeficientes de regressão. Prentice (1976) aponta dificuldades em estimar seu modelo com os dois parâmetros livres utilizando máxima verossimilhança e sugere fixar um deles e calcular o outro. Devido ao potencial destes modelos, foi feita a implementação deles utilizando tanto a abordagem frequentista quanto a Bayesiana para o pacote R (R Core Team, 2012).

O Capítulo 2 apresenta um retrospecto da regressão binária e define formalmente as funções de ligação. No Capítulo 3, mostra-se como as estimações são feitas nas duas teorias inferenciais e apresenta-se algumas funções de utilidade criadas para facilitar a visualização dos resultados. O Capítulo 4 é dedicado a exemplificar os modelos implementados comentando-os. Por fim, o Capítulo 5 apresenta as conclusões e propostas de trabalhos.

Capítulo 2

Regressão binária

A função logística foi criada no século XIX e era usada para modelar crescimento populacional (Cramer, 2002). A expressão $p = \dfrac{e^x}{1+e^x}$ surge como um modelo em que o crescimento é proporcional à população atual e a um máximo permitido. Já na época foi atribuído o nome que é usado hoje, mas apenas cerca de 80 anos depois é que ele se tornou corrente. Um problema desse modelo é que ele subestima a população máxima, apesar de funcionar bem antes de alcançar esse número. No começo do século XX, o modelo logístico foi amplamente utilizado para praticamente qualquer população, apesar dos problemas já conhecidos.

O termo probito* foi introduzido nos anos 30 por Bliss (1934) como um modelo para avaliar a resposta de indivíduos para certos estímulos. Ele mostra que para qualquer frequência relativa f, existe uma quantidade equivalente z tal que a função de distribuição normal no ponto z é igual a f. Com isso, era possível relacionar quantidades reais com a frequência relativa de ocorrência de eventos. A semelhança entre as distribuições normal e logística já era conhecida na época e a substituição da primeira pela segunda facilitava os cálculos feitos a mão.

Ambos os modelos pertencem à família dos modelos lineares generalizados (MLG) e é neste contexto que é feita a seguinte definição:

Definição 2.1 *Seja Y um vetor aleatório $n \times 1$ independente em que cada elemento segue distribuição Bernoulli de parâmetro π. Seja X uma matriz $n \times p$ de valores reais e β um vetor $p \times 1$ de coeficientes reais desconhecidos. Uma regressão binária é aquela que relaciona*

*Em inglês, *probit* é uma abreviação para *probability unit*.

a esperança de Y com a combinação linear de X e β como:

$$g(E(Y)) = \eta = X\beta,$$

para alguma função g inversível cujo domínio seja o intervalo $[0, 1]$ e a imagem seja os números reais.

A função g chama-se *função de ligação*, η é o *preditor linear* e os elementos de β são os coeficientes de regressão. A matriz X é chamada de *matriz de planejamento* e os vetores que a compõe são as *variáveis explicativas*. A primeira coluna de X é usualmente composta por um vetor $n \times 1$ de uns e o coeficiente de regressão associado a ela chama-se *intercepto*. Entende-se por *variável resposta* o vetor Y que representa a característica binária de interesse.

2.1 Funções de ligação

Sete funções de ligação serão apresentadas divididas em duas categorias. A denominação *paramétrica* é dada àquelas que possuem algum parâmetro para estimar além dos β da Definição 2.1. As ligações logito, probito e complementar log-log são encontradas em praticamente qualquer pacote estatístico. As outras quatro (Aranda-Ordaz, Weibull, Prentice e Stukel) são os modelos para os quais fez-se a implementação em R.

2.1.1 Logito

Essa é a função mais utilizada em regressão binária e também é chamada ligação canônica. Trata-se de uma ligação não paramétrica da seguinte forma:

$$g(\pi) = \ln\left(\frac{\pi}{1-\pi}\right)$$

e sua inversa:

$$g^{-1}(\eta) = \frac{\exp(\eta)}{1+\exp(\eta)}$$

É originada da distribuição logística padrão[†] e simétrica em torno de zero. Uma vantagem deste modelo é poder interpretar a exponencial dos coeficientes de regressão como razão de chances.

2.1.2 Probito

Essa é uma função não paramétrica da seguinte forma:

$$g(\pi) = \Phi^{-1}(\pi),$$

em que Φ é a função de distribuição da normal padrão.

Como a ligação logito, ela também é simétrica em torno de zero, porém não possui a interpretação dos parâmetros como razão de chances. Uma outra desvantagem é a inexistência de forma fechada para Φ, forçando a utilização de aproximações.

2.1.3 Complementar log-log

A ligação complementar log-log deriva da distribuição Valor (Mínimo) Extremo tipo I. Define-se a ligação:

$$g(\pi) = \ln(-\ln(1-\pi))$$

e sua inversa:

$$g^{-1}(\eta) = 1 - \exp(-\exp(\eta))$$

Também é uma ligação não paramétrica, mas esta não é simétrica e aproxima-se mais rápido de 1 conforme η aumenta do que as funções anteriores.

A Figura 2.1 ilustra as três funções de ligação apresentadas até agora. A complementar log-log é capaz de modelar melhor probabilidades próximas a 1, mas comporta-se como a logito para probabilidades baixas. A ligação probito é próxima da logito, mas aproxima-se dos extremos ligeiramente mais rápido.

[†]Ou seja, tem parâmetro de locação igual a zero e parâmetro de dispersão igual a um.

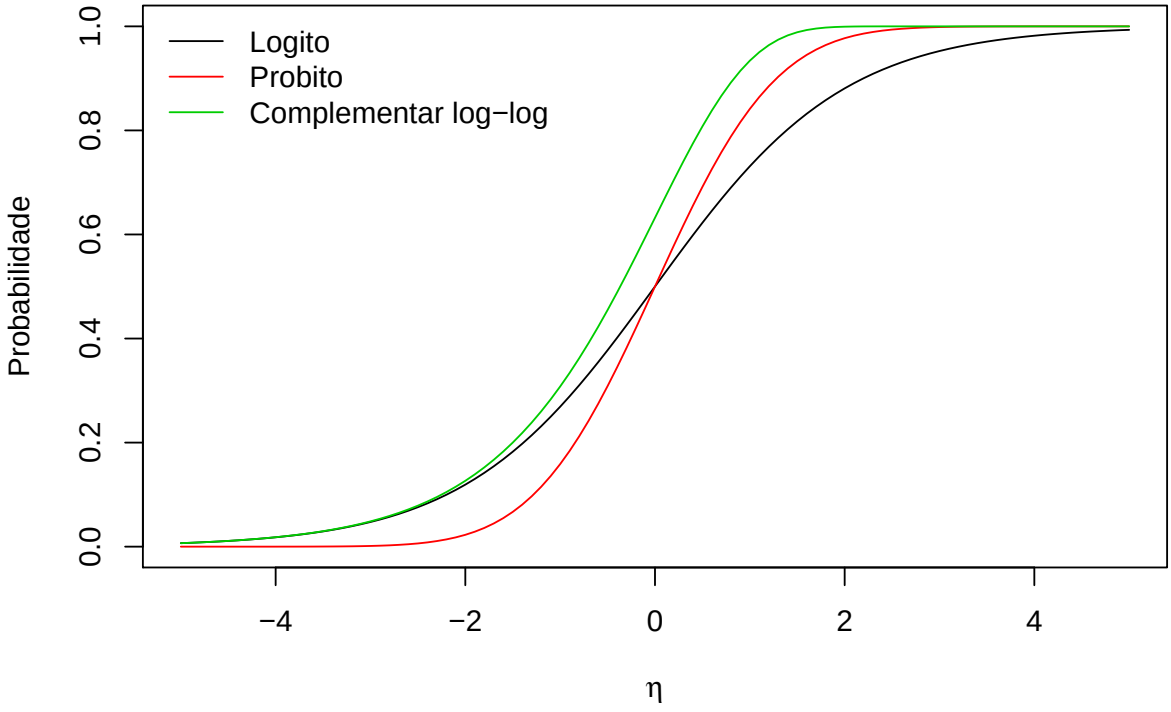

Figura 2.1: *Comparação das funções de ligação não paramétricas*

2.1.4 Aranda-Ordaz

Esta é a mais simples das ligações paramétricas discutidas, mas não é gerada a partir de uma distribuição de probabilidades. A ligação tem a seguinte forma:

$$g(\pi) = \ln\left(\frac{(1-\pi)^{-\alpha} - 1}{\alpha}\right)$$

e sua inversa:

$$g^{-1}(\eta) = \begin{cases} 1 - (\alpha \exp(\eta) + 1)^{-1/\alpha}, & \text{se } \alpha \exp(\eta) > -1 \\ 1, & \text{caso contrário} \end{cases}$$

Importante notar que a ligação só está definida para $\alpha > 0$. Dois casos particulares surgem ao tomar $\alpha = 1$ em que a função reduz-se à logito e quando $\alpha \to 0$, obtém-se a ligação complementar log-log.

A Figura 2.2 mostra o comportamento da função para alguns valores de α. A simetria ocorre apenas quando $\alpha = 1$. Conforme α aumenta, a função aproxima-se mais lentamente de 1 e valores baixos do parâmetro têm o efeito contrário. Para proporções próximas a zero o valor de α quase não influencia, sendo todas as curvas muito semelhantes.

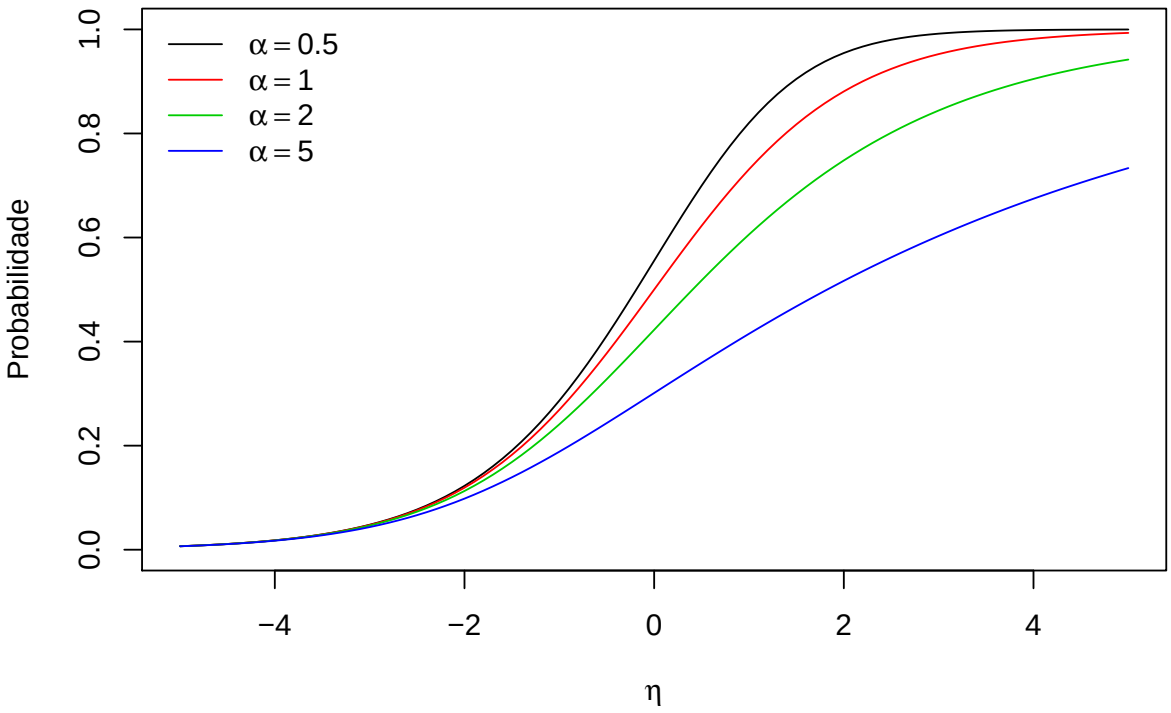

Figura 2.2: *Ligação Aranda-Ordaz para alguns valores de* α

2.1.5 Weibull

A mais recente das funções estudadas, surge a partir da distribuição Weibull de três parâmetros. Define-se a ligação por:

$$g(\pi) = [-\ln(1-\pi)]^{1/\gamma}$$

e sua inversa:

$$g^{-1}(\eta) = 1 - \exp(-\eta^{\gamma})$$

O parâmetro γ deve ser positivo como na distribuição Weibull. A ligação equivale-se à complementar log-log quando $\gamma \to \infty$ e tem como casos limite tanto a ligação logito quando a probito (Caron, 2010).

A Figura 2.3 mostra a forma da função de ligação para alguns valores de γ. Ao contrário da ligação Aranda-Ordaz, valores altos do parâmetro aproximam a função da proporção 1 mais rapidamente, mas diferente da primeira, é possível obter diferentes comportamentos da cauda esquerda modificando-se o valor do parâmetro.

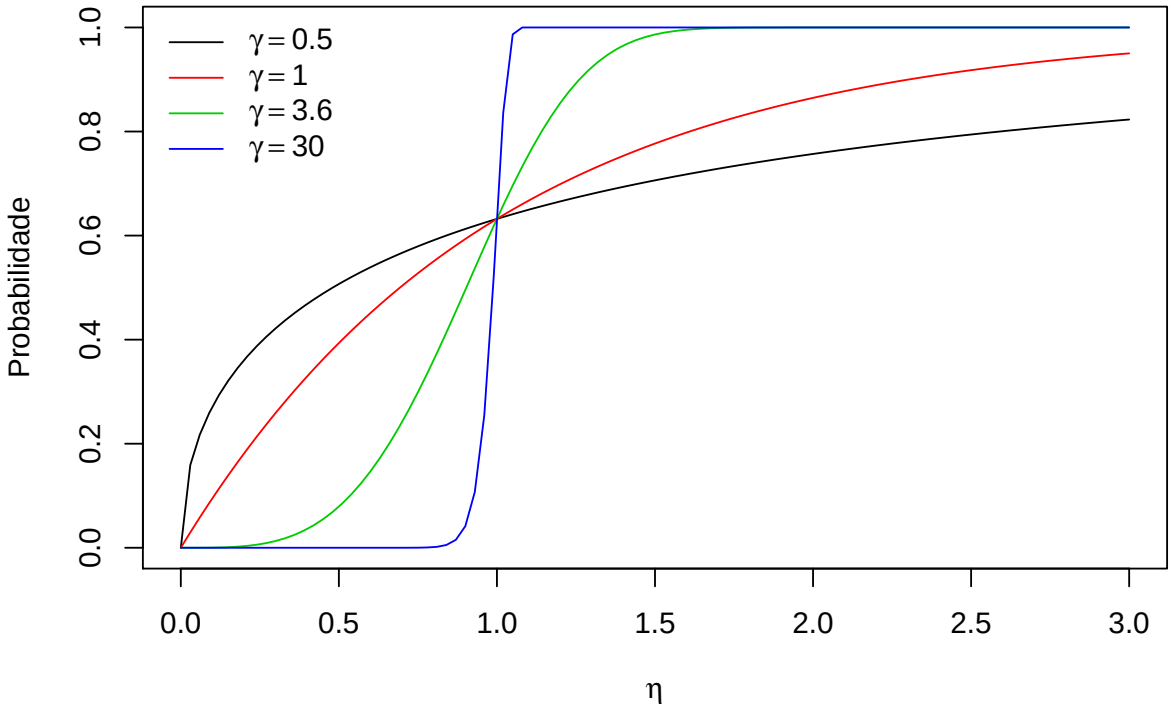

Figura 2.3: *Ligação Weibull para alguns valores de* γ

2.1.6 Prentice

A ligação sugerida por Prentice (1975) surge ao tomar o logaritmo de uma variável aleatória com distribuição F-Snedecor de parâmetros $2m_1$ e $2m_2$. O cálculo da função de ligação e de sua inversa será discutido no Capítulo 3.

Casos particulares dessa distribuição incluem a logística ($m_1 = m_2 = 1$), valor mínimo extremo ($m_1 = 1, m_2 \to \infty$), valor máximo extremo ($m_1 \to \infty, m_2 = 1$), normal degenerada

$(m_1 \to \infty, m_2 \to \infty)$, exponencial $(m_1 \neq 0, m_2 \to 0)$ e exponencial dupla $(m_1 \to 0, m_2 \to 0)$. Também é importante notar que a distribuição é simétrica para $m_1 = m_2$.

A Figura 2.4 mostra a forma da ligação para alguns valores do vetor $m = (m_1, m_2)$. Os parâmetros permitem controlar as caudas da função, sendo o primeiro responsável pela cauda esquerda e o segundo, pela direita. Quanto maior o valor de m_1, mais rapidamente a função irá se aproximar de 0 e o mesmo ocorre com m_2, mas com a função aproximando-se de 1.

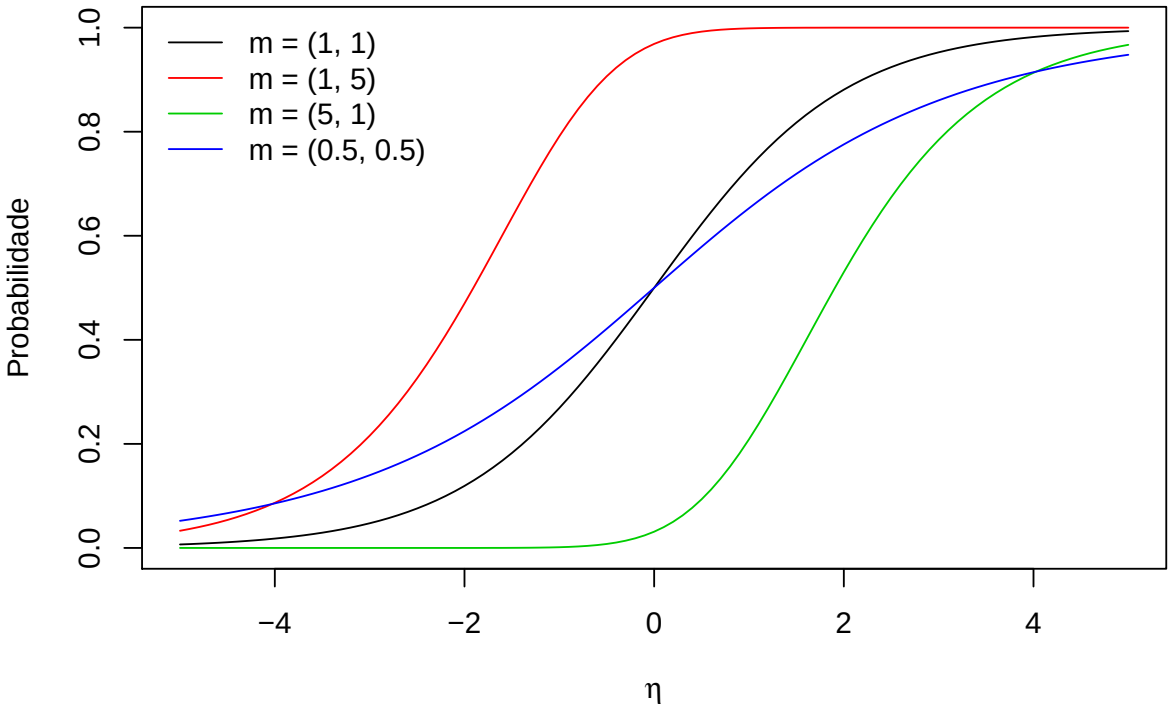

Figura 2.4: *Ligação Prentice para alguns valores de m*

2.1.7 Stukel

Por fim, a ligação de Stukel (1988) compreende uma generalização da logito em que uma função h será aplicada ao preditor linear antes da ligação logito. Essa função tem a seguinte forma:

$$h_a(\eta) = \begin{cases} \dfrac{\exp(a_1\eta) - 1}{a_1}, & \text{se } a_1 > 0 \text{ e } \eta > 0 \\ \eta, & \text{se } a_1 = 0 \text{ e } \eta > 0 \\ -\dfrac{\ln(1 - a_1\eta)}{a_1}, & \text{se } a_1 < 0 \text{ e } \eta > 0 \\ -\dfrac{\exp(-a_2\eta - 1)}{a_2}, & \text{se } a_2 > 0 \text{ e } \eta \leq 0 \\ \eta, & \text{se } a_2 = 0 \text{ e } \eta \leq 0 \\ \dfrac{\ln(1 + a_2\eta)}{a_2}, & \text{se } a_2 < 0 \text{ e } \eta \leq 0 \end{cases}$$

A função de ligação é então definida da seguinte maneira:

$$g(\pi) = h_a^{-1}\left(\ln\left(\frac{\pi}{1-\pi}\right)\right)$$

e sua inversa:

$$g^{-1}(\eta) = \frac{\exp(h_a(\eta))}{1 + \exp(h_a(\eta))}$$

Assim como na ligação Prentice, há simetria quando $a_1 = a_2$ e o modelo logito é um caso particular se ambos os parâmetros forem nulos. Ainda é possível aproximar outras ligações como a probito ($a_1 = a_2 \approx 0{,}165$) e a complementar log-log ($a_1 \approx 0{,}62$ e $a_2 \approx -0{,}037$).

Os dois parâmetros governam as caudas de forma independente como é possível ver na definição da função h e na Figura 2.5, que apresenta a forma da ligação para alguns valores selecionados do vetor $a = (a_1, a_2)$. Quanto maior o valor do parâmetro, mais rapidamente a curva se aproxima do extremo, ou seja, para proporção 1 no caso de a_1 e para proporção 0 no caso de a_2.

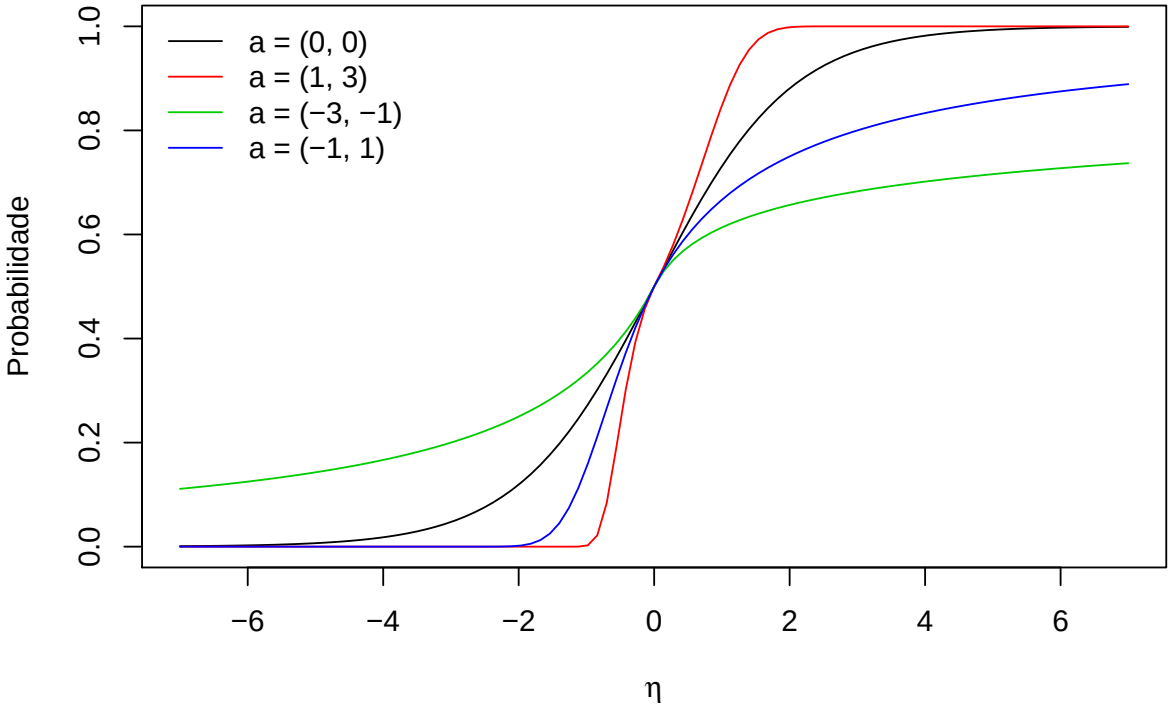

Figura 2.5: *Ligação Stukel para alguns valores de a*

Capítulo 3

Algoritmo

Escolhidas as variáveis de interesse, o problema inicial de uma análise de regressão consiste em estimar os parâmetros do modelo. Há duas teorias inferenciais que regem a maneira como os coeficientes serão estimados: frequentista (ou clássica) e Bayesiana. Neste trabalho, considera-se o método frequentista de estimação por maximização da função de verossimilhança. Este método é bem difundido e o utilizado nos pacotes estatísticos para a estimação das funções de ligação logito, probito e complementar log-log. De maneira geral, ele não possui forma fechada para os estimadores, necessitando de métodos numéricos de otimização como o de Newton-Raphson (Dennis e Schnabel, 1996). Métodos Bayesianos também são baseados na função de verossimilhança, mas também dependem das distribuições *a priori* dos parâmetros que devem ser fornecidas pelo pesquisador. A distribuição *a posteriori* não costuma ter forma fechada para modelos mais complexos do que a regressão linear normal, sendo necessário métodos numéricos para gerar uma amostra desta distribuição.

Este capítulo apresenta algoritmos em R (R Core Team, 2012) para a estimação de modelos de regressão binária com as funções de ligação paramétricas apresentadas no Capítulo 2 em ambas teorias de inferência. Primeiramente será apresentado o cálculo das funções de verossimilhança comum às duas abordagens. As duas seções seguintes propõem uma solução do problema e sua implementação. O código completo pode ser encontrado no Apêndice A e está disponível em `http://dl.dropbox.com/u/8396505/binreg.R`.

3.1 Função de verossimilhança

O primeiro passo do problema consiste em calcular a função de verossimilhança:

$$L(\theta \,|\, Y, X) = \prod_{i=1}^{n} P(Y_i = y_i \,|\, \theta, X_i),$$

em que Y_i segue uma distribuição Bernoulli, n é o número de observações, $Y = (y_1, \ldots, y_n)$, $X = (X_1, \ldots, X_n)$, y_i e X_i são as variáveis resposta e explicativas, respectivamente, da i-ésima observação e θ é o vetor de parâmetros a ser estimado consistindo de todos os coeficientes de regressão mais os parâmetros da função de ligação. De maneira geral, a função terá a seguinte forma:

$$L(\theta \,|\, Y, X) = \prod_{i=1}^{n} (g^{-1}(\eta_i))^{y_i} (1 - g^{-1}(\eta_i))^{1-y_i},$$

ou ainda, a função de log-verossimilhança que será de fato utilizada:

$$l(\theta \,|\, Y, X) = \sum_{i=1}^{n} y_i \ln(g^{-1}(\eta_i)) + \sum_{i=1}^{n} (1 - y_i) \ln(1 - g^{-1}(\eta_i))$$

O objetivo principal aqui é programar a função g^{-1}. A ligação Aranda-Ordaz é a mais simples e basta utilizar diretamente a definição da Seção 2.1.4. A ligação Weibull também não apresenta dificuldades, exceto pelo fato do suporte da distribuição Weibull ser o intervalo $(0, \infty)$. Neste caso, o intercepto da regressão linear atua como um parâmetro de locação, fazendo com que o preditor linear η seja sempre positivo. No processo de estimação, propõe-se o uso de $\eta_+ = \max(\eta, 0)$, isto é, um truncamento de η em zero. A ligação Stukel também apresenta pouca dificuldade de implementação. Diversas declarações *if* aninhadas dão conta da programação da função h e depois basta aplicar a função logística ao resultado.

O maior problema está em calcular as funções de probabilidade acumulada e quantil da distribuição definida por Prentice. Uma solução, no entanto, pode ser encontrada no Apêndice 1 de Prentice (1976) onde se lê:

The response probability (1), with underlying density given by (2), can be written

$$I(z; m_1, m_2)/\beta(m_1, m_2),$$

where $z = \exp y/(1 + \exp y)$ *and* I *represents the incomplete* β *integral. For* $z \leq .5$ *this integral was calculated (Osborn e Madey, 1968) as*

$$z^{m_1} \sum_{j=0}^{\infty} A_j/(m_1 + j)$$

where $A_0 = 1$, *and* $A_j = \{(j - m_2)z/j\}A_{j-1}, j = 1, 2, \cdots$, *with summation terminating when the relative contribution of the last calculated term was less than* 10^{-7}. *For* $.5 < z \leq 1$ *the relation*

$$I(z; m_1, m_2) = \beta(m_1, m_2) - I(1 - z; m_2, m_1)$$

was utilized. The complete β *integral was calculated as* $I(.5, m_1, m_2) + I(.5, m_2, m_1)$. *The incomplete normal integral was approximated as suggested in Abramowitz e Stegun (1965, p. 932, expression 26.2.19).*

A equação (1) citada define a inversa da função de ligação e (2) é densidade do logaritmo da distribuição F-Snedecor de parâmetros $2m_1$ e $2m_2$. Este trecho mostra que esta função de distribuição pode ser escrito como a função de distribuição de uma variável aleatória com distribuição Beta(m_1, m_2) aplicada no ponto $p = \dfrac{e^x}{1 + e^x}$. O algoritmo utilizado para a função beta incompleta foi o encontrado no sistema base do R ao invés do sugerido no artigo supracitado. A função quantil será necessária na próxima seção, porém basta inverter a relação acima que o resultado será a função logística aplicada sobre a função quantil de uma distribuição Beta(m_1, m_2).

Há, ainda, três pontos que devem ser notados na implementação das funções de log-verossimilhança. O primeiro é que as funções retornam $-l(\theta \,|\, Y, X)$, pois a otimização que será utilizada na abordagem frequentista executa apenas minimização. A segunda observação é que deve-se utilizar um algoritmo de maximização com restrição dada pelo espaço paramétrico das funções de ligação. Para isso, faz-se o valor de $l(\theta \,|\, Y, X) = -10^{100}$ quando

$\theta \notin \Theta$, em que Θ é o espaço paramétrico. Por último, a fim de evitar proporções numericamente nulas ou iguais a um, após aplicada a função de ligação no preditor linear, todos os valores inferiores a $\epsilon = 2{,}220446 \times 10^{-16*}$ são truncados em ϵ e o análogo ocorre quando superiores a $1-\epsilon$. Desta forma evita-se que, no cálculo da função de verossimilhança, algum dos fatores seja $\ln(0)$, o que impossibilitaria a estimação dos modelos. Este procedimento é usualmente adotado nos pacotes estatísticos.

3.2 Abordagem frequentista

Parâmetros na função de ligação adicionam um desafio no método de estimação. Aranda-Ordaz (1981); Prentice (1975, 1976); Stukel (1988) não apresentam uma forma de estimar seus modelos considerando todos os parâmetros da função de ligação simultaneamente com os parâmetros de regressão linear. Prentice (1975, 1976); Stukel (1988) apresentam exemplos estimando apenas um dos parâmetros da função de ligação, enquanto que Aranda-Ordaz (1981) não estima o parâmetro da ligação, ajustando o modelo para apenas alguns valores de α fixos. Inclusive, entende-se que este seja o motivo pelo qual nenhum pacote estatístico possua uma rotina para a estimação destes modelos.

Desta forma, para a estimação dos modelos apresentados, considerou-se o método da máxima verossimilhança perfilada (Fisher, 1956), estimando-se os parâmetros da função de ligação separadamente dos parâmetros de regressão linear num processo iterativo até a obtenção final dos estimadores. De forma geral, o algoritmo proposto é dado por:

1. Dados valores iniciais para os parâmetros de ligação, estimar os coeficientes de regressão como um MLG usual;

2. Fixado os coeficientes de regressão do passo anterior, estimar os parâmetros de ligação maximizando a verossimilhança condicional;

3. Fixado os parâmetros de ligação, reajustar os coeficientes de regressão;

4. Repetir os passos 2 e 3 até que as contribuições relativas à iteração anterior sejam menor que uma tolerância ou um número máximo de iterações seja atingido.

*Menor número que o R versão 2.15.1 reconhece

Essa contribuição pode ser tanto sobre a estimativa dos parâmetros ou sobre o AIC (Akaike, 1974), o que ocorrer primeiro. Dessa maneira, o parâmetro pode não ter sido estabilizado ainda, mas se dar mais um passo de estimação não melhora o modelo além da tolerância, é considerado que não vale a pena o tempo de processamento extra. A tolerância padrão foi fixada em 10^{-4} após testes com diversos valores. Os ganhos obtidos com tolerâncias maiores sempre foram menores do que se esperaria com tempo de execução substancialmente maior. De toda maneira, há uma opção para o usuário modificar para o valor que preferir. O número máximo de iterações padrão é de 5000 passos; este valor é pelo menos cinco vezes o número máximo de iterações necessárias para o algoritmo parar devido tolerância padrão ser atingida nos testes executados.

O algoritmo proposto tem duas vantagens principais: 1. os resultados passam no teste de sanidade com relação à matriz de informação observada; e 2. o objeto retornado será uma chamada a `glm`, então todas as funções para análise de resultados como `predict`, `resid`, `drop1`, entre outras podem ser utilizadas diretamente.

3.2.1 Código comentado

Esta seção abordará toda a parte de programação necessária para rodar o algoritmo descrito em linguagem R. Como exemplo será utilizado o modelo de Prentice por apresentar maiores dificuldades; os outros são análogos.

O primeiro passo é definir uma função de ligação customizada. Isso é feito declarando uma função que retorna uma lista de funções, definindo a função de ligação, a inversa, a derivada da inversa com relação a η, um texto para identificar a função e eventuais restrições a η (Código 3.1).

```
1    prentice <- function(m = c(1, 1)) {
2      linkfun <- function(mu)  qprentice(mu,m)
3      linkinv <- function(eta) pprentice(eta,m)
4      mu.eta <- function(eta) dprentice(eta,m)
5      link <- paste('Prentice, m1 = ', m[1], '  -- m2 = ', m[2], sep = '')
6      valideta <- function(eta) TRUE
7      structure(list(linkfun = linkfun, linkinv = linkinv,
8              mu.eta = mu.eta, valideta = valideta, name = link),
```

```
9                 class = 'link-glm')
10    }
```

Código 3.1: *Definição da ligação de Prentice*

Isso permite utilizar o modelo de Prentice com valores predefinidos de m como, por exemplo, `glm(formula, binomial(prentice(c(0.5, 1))))`. Vale ressaltar o uso das funções `qprentice`, `pprentice` e `dprentice` que devem ser definidas como no Código 3.2 a seguir:

```
1     pprentice <- function(q, m = c(1, 1)) {
2       if(length(m) < 2)
3         stop('m = ', deparse(substitute(m)), ' should have exactly two
              elements.')
4       if(length(m) > 2)
5         warning('Length of m = ', deparse(substitute(m)), 'is greater
              than 2. Only the first two elements will be used.')
6
7       return( pbeta(plogis(q), m[1], m[2]) )
8     }
9
10    qprentice <- function(p, m = c(1, 1)) {
11      if(length(m) < 2)
12        stop('m = ', deparse(substitute(m)), ' should have exactly two
              elements.')
13      if(length(m) > 2)
14        warning('Length of m = ', deparse(substitute(m)), 'is greater
              than 2. Only the first two elements will be used.')
15
16      return( qlogis(qbeta(p, m[1], m[2])) )
17    }
18
19    dprentice <- function(x, m = c(1, 1)) {
20      if(length(m) < 2)
21        stop('m = ', deparse(substitute(m)), ' should have exactly two
              elements.')
22      if(length(m) > 2)
23        warning('Length of m = ', deparse(substitute(m)), 'is greater
```

```
            than 2. Only the first two elements will be used.')
24
25      exp(x*m[1])*((1 + exp(x))^(-(m[1] + m[2])))/beta(m[1], m[2])
26    }
```

Código 3.2: *Funções distribuição, quantil e densidade da distribuição log F*

A maneira de calcular essas funções com base na distribuição Beta já foi discutida anteriormente. Note que há uma pequena verificação de erro para o vetor m. Ele deve ter exatamente dois elementos; caso haja apenas um, a execução é interrompida com erro, mas se há mais de dois, apenas os primeiros são usados e uma mensagem de aviso é retornada.

Também deve ser definida a função de menos log-verossimilhança a ser minimizada como no Código 3.3:

```
1   likprentice <- function(m = c(1, 1), beta, y, X){
2     if(all(m > 0)){
3       eta <- X %*% beta
4       prentice <- pprentice(eta, m)
5       prentice <- pmin(pmax(prentice, .Machine$double.eps), 1 -
            .Machine$double.eps)
6       return(-sum(dbern(y, prentice, log = TRUE)))
7     } else {
8       return(1e100)
9     }
10  }
```

Código 3.3: *Log-verossimilhança para o modelo de Prentice*

Note como proporções numericamente iguais a 1 ou 0 são passadas para o número mais próximo possível que a linguagem reconhece após aplicar a inversa da ligação. Também ver que é retornado menos a função de log-verossimilhança e o uso da função `dbern` ao invés de `dbinom`. A primeira função não está no pacote básico do R, mas sim em um que será usado para a estimação Bayesiana, mas testes mostraram que ela roda ligeiramente mais rápida do que a básica, justificando sua escolha. Importante, também, é que se os parâmetros de ligação não forem todos positivos, a função retorna um número muito alto para que a escolha desses candidatos seja descartada ao minimizar a menos verossimilhança.

A função principal que será chamada pelo usuário tem a seguinte forma:

```
binreg(formula, link = c('Aranda-Ordaz', 'Weibull', 'Prentice', 'Stukel'),
          data, subset, start = NULL, tol = 1e-4, iterlim = 5000, na.action)
```

O primeiro argumento deve ser uma fórmula do R do tipo `resposta ~ x1 + x2` em que `x1` e `x2` são variáveis explicativas. O próximo argumento é uma *string* que define a função de ligação. O modelo de Aranda-Ordaz é o padrão que será rodado caso nenhuma seja especificada e, também, não é necessário escrever o nome inteiro porque está habilitada uma opção de comparação parcial. De fato, basta a primeira letra, já que todos começam com uma diferente. Os argumentos `data` e `subset` definem o objeto com os dados e eventuais restrições. Em `start` pode ser passado um vetor inicial de coeficientes de regressão. Este deve conter exatamente o mesmo número de termos que serão estimados. Os próximos dois argumentos já foram discutidos, mas eles controlam a tolerância do critério de parada (`tol`) e o número máximo de iterações (`iterlim`). Por fim, é possível modificar o tratamento dado a valores faltantes; se nada for declarado, o valor em `options()$na.action` será utilizado.

Antes de executar o algoritmo descrito na Seção 3.2, é preciso fazer algumas verificações de erro ou utilização de valores padrão. A primeira parte da função `binreg` serve justamente para esse propósito. Um teste é feito para se ter certeza de que o argumento passado como `formula` realmente é uma fórmula e outro para ver se é possível encontrar uma função de ligação com o nome que foi digitado. Os argumentos `data` e `subset` não são obrigatórios, mas quando o primeiro não está presente, as variáveis utilizadas devem estar disponíveis no mesmo ambiente da fórmula (geralmente com o comando `attach` ou declaradas fora de um `data.frame`).

Outro passo é determinar a matriz de planejamento e a variável resposta com os comandos `model.matrix` e `model.frame`, respectivamente. É importante notar que a variável passada no lado esquerdo da fórmula será transformada em categórica, sendo o primeiro nível considerado fracasso e os demais, sucesso. Este é o mesmo comportamento adotado pelo `glm` na família `binomial`. Se o vetor de respostas tiver menos que dois valores distintos, a execução é interrompida com uma mensagem (Código 3.4).

```
1   call <- match.call()
2   if(class(formula) != 'formula')
3     stop('Invalid formula')
4   if(missing(data))
5     data <- environment(formula)
6   if(!missing(subset))
7     data <- subset(data, subset)
8   X    <- model.matrix(formula, data)
9   resp <- model.frame(formula, data)[,1]
10  resp <- factor(resp)
11
12  if(nlevels(resp) > 2)
13    warning('Number of levels greater than 2. The first one will be
          regressed against the others.')
14  if(nlevels(resp) < 2)
15    stop('Response has fewer than two levels.')
16  resp <- ifelse(resp == levels(resp)[1], 0, 1)
17
18  link <- match.arg(link)
```

Código 3.4: *Verificações iniciais da função binreg*

Finalmente é possível começar a estimação de fato. Primeiro é necessário inicializar uma variável para contar os passos e outra para controlar o AIC do modelo. Uma série de *ifs* escolhe qual modelo ajustar, mas, novamente, apenas o de Prentice será comentado. Antes de iniciar o processo iterativo, algumas definições iniciais precisam ser feitas. No exemplo do Código 3.5, inicia-se o vetor $m = (1, 1)$, ou seja, equivalente ao modelo logístico, calcula-se os coeficientes de regressão iniciais com uma chamada a `glm` e a menos log-verossimilhança é minimizada com a função `nlm` dados os parâmetros já calculados. Nesse caso específico (e também para o modelo Weibull), há uma variável especial chamada *flag* que é utilizada para verificar se os parâmetros não estão crescendo incessantemente. Caso em 10 iterações seguidas ambos os parâmetros sempre aumentarem de valor e a contribuição no AIC for muito pequena, será considerado que eles vão continuar com essa tendência e um modelo probito é retornado no lugar, assumindo que $m \to (\infty, \infty)$.

```
1    i    <- 0
2    aic  <- 0
3
4    if(link == 'Prentice'){
5      m    <- c(1, 1)
6      fit  <- glm(formula, binomial(prentice(m)), data, start = start)
7      bet  <- coef(fit)
8      cand <- nlm(likprentice, m, beta = bet, y = resp, X = X)
9      flag <- c(0.1, 0.1)
10
11     while(((any(abs(cand$estimate - m) > tol)) ||
           (any((abs(cand$estimate - m)/m) > tol))) && (abs(fit$aic - aic)
           > tol)){
12       if ((i >= 10) && (abs(fit$aic - aic) < tol) &&
13         ((all(flag[(length(flag)-10):length(flag),1] > 0)) &&
14         (all(flag[(length(flag)-10):length(flag),2] > 0)))){
15         warning('Link parameters are too large. Probit estimate
               returned.')
16         fit <- glm(formula, binomial('probit'), data, na.action =
               na.action)
17         fit$formula <- formula
18         fit$call <- call
19         return(fit)
20       }
21
22       aic  <- fit$aic
23       m    <- cand$estimate
24       fit  <- glm(formula, binomial(prentice(m)), data, start = bet)
25       bet  <- coef(fit)
26       cand <- nlm(likprentice, m, beta = bet, y = resp, X = X, hessian
           = TRUE)
27       i    <- i + 1
28       flag <- rbind(flag, (cand$estimate - m))
29
30       if(i == iterlim){
```

```
31          warning('Maximum iteration reached. Solution may not be
                optimal.')
32          break
33        }
34      }
35
36      m <- cand$estimate
37      fit <- glm(formula, binomial(prentice(m)), data, start = bet,
            na.action = na.action)
38      fit$link.par <- m
39      fit$link.err <- as.numeric(sqrt(diag(solve(cand$hessian))))
40      names(fit$link.par) <- names(fit$link.err) <- c('m1', 'm2')
41      fit$aic <- fit$aic + 4
42    }
```

Código 3.5: *Estimação de parâmetros do modelo Prentice*

O critério de parada do laço *while* é se as mudanças nos parâmetros ou no AIC são menores do que a tolerância e logo após é feita a verificação da *flag*. Caso tudo isso seja negativo, então novos valores dos coeficientes de regressão e dos parâmetros de ligação são estimados com `glm` e `nlm`. No final há uma verificação do número de iterações e caso o máximo seja atingido, a função retorna os valores atuais com um aviso de que a estimativa pode não ser a ótima.

Terminadas as iterações, um último ajuste de coeficientes de regressão é feito e os erros-padrão são calculados a partir da matriz de informação de Fisher observada. Também é feito um ajuste no AIC do modelo para incorporar os parâmetros extras calculados. Um último passo consiste em corrigir os campos de chamada (`call`), fórmula, (`formula`) e classe (`class()`) além de incluir um novo com o número de iterações.

```
1   fit$iterations <- i
2   fit$formula <- formula
3   fit$call <- call
4   class(fit) <- c('binreg', class(fit))
5   fit
```

Código 3.6: *Correções finais*

3.2.2 Métodos para classe binreg

Os códigos apresentados resolvem o problema da estimação dos parâmetros e calcula todos os erros-padrão. Como a saída é um objeto de classe `glm`, todas as funções que têm métodos para essa classe funcionam, mas algumas necessitam de pequenos ajustes. Um dos propósitos do Código 3.6 é justamente criar uma classe nova para acomodar essas alterações.

Uma preocupação sobre funções de ligações paramétricas é que o R não inclui seus parâmetros na conta dos graus de liberdade. Isso causa erros nos cálculos de critério de informação como AIC e BIC (Schwarz, 1978) pois estes dependem dessa quantidade. A documentação do R deixa claro que basta criar métodos para a função `logLik` corrigindo os graus de liberdade que o problema é resolvido (Código 3.7).

```
1   logLik.binreg <- function(object, ...){
2     if (length(list(...)))
3       warning('extra arguments discarded')
4
5     link <- strtrim(object$family$link, 1)
6     p <- object$rank + switch(link, A = 1, W = 1, P = 2, S = 2)
7
8     val <- p - object$aic/2
9     attr(val, 'nobs') <- sum(!is.na(object$residuals))
10    attr(val, 'df') <- p
11    class(val) <- 'logLik'
12    val
13  }
```

Código 3.7: *Função de log-verossimilhança para classe binreg*

Outras duas funções para a classe `binreg` foram modificadas das versões para classe `glm` para mostrar os parâmetros de ligação. No caso da função `print` que mostra os resultados resumidamente, apenas as estimativas são apresentadas (Código 3.8).

```
1   print.binreg <- function (x, digits = max(3, getOption('digits') -
        3), ...){
2     cat('\nCall:  ', paste(deparse(x$call), sep = '\n', collapse =
        '\n'), '\n\n', sep = '')
3     if (length(coef(x))) {
```

```
4         cat('Coefficients')
5         if (is.character(co <- x$contrasts))
6           cat('  [contrasts: ', apply(cbind(names(co), co), 1L, paste,
                collapse = '='), ']')
7         cat(':\n')
8         print.default(format(x$coefficients, digits = digits), print.gap
              = 2, quote = FALSE)
9       }
10      else cat('No coefficients\n\n')
11
12      cat('\nLink Parameters:\n')
13      print.default(format(x$link.par, digits = digits), print.gap = 2,
            quote = FALSE)
14
15      cat('\nDegrees of Freedom:', x$df.null, 'Total (i.e. Null); ',
            x$df.residual, 'Residual\n')
16      if (nzchar(mess <- naprint(x$na.action)))
17        cat('  (', mess, ')\n', sep = '')
18      cat('Null Deviance:\t    ', format(signif(x$null.deviance,
19            digits)), '\nResidual Deviance:', format(signif(x$deviance,
20            digits)), '\tAIC:', format(signif(x$aic, digits)), '\n')
21      invisible(x)
22    }
```

Código 3.8: *Função para impressão de resultados para classe binreg*

Um método para `summary` também foi escrito para comportar os parâmetros de ligação. Estes apareceram sempre após os coeficientes de regressão. Aqui é importante apresentar o teste de Wald (Agresti, 2002) para os parâmetros de ligação, sendo a hipótese nula sempre os valores que fazem o modelo ser reduzido a uma regressão logística como descrito na Seção 2.1 (Código 3.9).

```
1   summary.binreg <- function(object, ...){
2     ans <- NextMethod('summary')
3     link <- strtrim(object$family$link, 1)
4
5
```

```
6       if(link == 'A'){
7         zval <- (object$link.par - 1)/object$link.err
8         pval <- pchisq(zval^2, 1, lower.tail = FALSE)
9         ans$coefficients <- rbind(ans$coefficients, alpha =
              c(object$link.par, object$link.err, zval, pval))
10      }
11
12      if(link == 'W'){
13        zval <- (object$link.par - 3.50215)/object$link.err
14        pval <- pchisq(zval^2, 1, lower.tail = FALSE)
15        ans$coefficients <- rbind(ans$coefficients, gamma =
              c(object$link.par, object$link.err, zval, pval))
16      }
17
18      if(link == 'P'){
19        zval <- (object$link.par - 1)/object$link.err
20        pval <- pchisq(zval^2, 1, lower.tail = FALSE)
21        ans$coefficients <- rbind(ans$coefficients, m1 =
              c(object$link.par[1], object$link.err[1], zval[1], pval[1]),
22              m2 = c(object$link.par[2], object$link.err[2], zval[2],
                    pval[2]))
23      }
24
25      if(link == 'S'){
26        zval <- object$link.par/object$link.err
27        pval <- pchisq(zval^2, 1, lower.tail = FALSE)
28        ans$coefficients <- rbind(ans$coefficients, a1 =
              c(object$link.par[1], object$link.err[1], zval[1], pval[1]),
29              a2 = c(object$link.par[2], object$link.err[2], zval[2],
                    pval[2]))
30      }
31
32      return(ans)
33    }
```

Código 3.9: *Função de resumo dos resultados para a classe binreg*

3.2.3 Funções extras

Uma alternativa importante para o teste de Wald é o teste da razão de verossimilhanças para avaliar se o modelo logístico se ajusta melhor aos dados. Assim, adota-se as hipóteses:

H_0 : Modelo logístico

H_1 : Modelo superior ao logístico

Seja a seguinte quantidade

$$\lambda = \frac{L(\theta_0 \,|\, X, Y)}{L(\theta_1 \,|\, X, Y)},$$

em que $L(\theta_0 \,|\, X, Y)$ representa a verossimilhança do modelo logístico com as mesmas variáveis explicativas do modelo a ser testado e $L(\theta_1 \,|\, X, Y)$ representa a verossimilhança do modelo de regressão binária de interesse. Note que o teste da razão de verossimilhanças é válido apenas para modelos encaixados, mas como visto no Capítulo 2, pode-se considerar que o modelo logito é um caso particular de todos os modelos aqui implementados. Então a estatística de teste $\text{LR} = -2\ln(\lambda)$ tem o seguinte resultado assintótico (Wilks, 1938):

$$-2\ln(\lambda) \overset{n\to\infty}{\sim} \chi^2_p,$$

em que p é o número de parâmetros na ligação.

O Código 3.10 executa esse procedimento para um modelo de classe `binreg` e retorna um objeto de classe `htest` próprio para testes de hipóteses.

```
1   LRTlogit <- function(object){
2     if(!inherits(object, 'binreg'))
3       stop('The object is not from class binreg')
4
5     dname <- deparse(substitute(object))
6     logit <- glm(object$formula, binomial, object$data)
7     lr <- 2*as.numeric(logLik(object) - logLik(logit))
8     df <- attr(logLik(object), 'df') - attr(logLik(logit), 'df')
9     pval <- pchisq(lrt, df, lower.tail = FALSE)
10    method <- 'Likelihood ratio test for Binary Regression with
          parametric link'
11
```

```
12      RVAL <- list(statistic = c(LR = lr), parameter = c(df = df),
            p.value = pval, method = method, data.name = dname)
13      class(RVAL) <- 'htest'
14      return(RVAL)
15    }
```

Código 3.10: *Teste da razão de verossimilhanças contra o modelo logístico*

O Código 3.11 implementa última função extra que funciona como um resumo dos critérios de informação para quantos modelos forem passados. Basicamente mostra a log-verossimilhança, graus de liberdade, AIC e BIC para todos os modelos numa forma fácil de comparar.

```
1   compareIC <- function(...){
2     names <- as.character(match.call())[-1]
3
4     objects <- list(...)
5
6     df <- sapply(objects, function(x) attr(logLik(x), 'df'))
7     llik <- sapply(objects, logLik)
8     aic <- sapply(objects, AIC)
9     bic <- sapply(objects, BIC)
10
11    data.frame(df = df, logLik = llik, AIC = aic, BIC = bic, row.names
          = names)
12  }
```

Código 3.11: *Função para comparar critérios de informação*

3.3 Abordagem Bayesiana

A inferência Bayesiana, ao contrário da frequentista, não utiliza apenas a função de verossimilhança para tomar decisões. A opinião do pesquisador também é levada em conta na forma de distribuições *a priori*. O Teorema de Bayes, então, é utilizado para combinar as duas fontes de informação gerando a distribuição *a posteriori*.

Nesse tipo de inferência os parâmetros a serem estimados são variáveis aleatórias e o primeiro grande problema é definir suas distribuições. Como estamos falando de coeficientes de regressão e parâmetros de ligação, é difícil ter uma ideia de seus comportamentos sem um estudo anterior e a grande gama de possíveis distribuições é outro problema. Assim, é muito difícil escrever um programa que dê todas as opções para o usuário sem complicar a interface. Por esse motivo, foi decidido que os parâmetros contínuos com espaço paramétrico definido na reta real teriam distribuição *a priori* normal, enquanto que os que só podem ser positivos têm *priori* normal positiva. Os parâmetros dessas distribuições podem ser modificados na própria função, mas o padrão será aqueles que aproximam de uma *priori* não informativa.

O segundo grande problema da inferência Bayesiana está no cálculo da distribuição *a posteriori*. Para visualizar o problema, defina $p(\theta)$ a função densidade *a priori* e $p(\theta \mid X, Y)$ a função densidade *a posteriori*. Com isso, o Teorema de Bayes diz que:

$$p(\theta \mid X, Y) = \frac{L(\theta \mid X, Y)p(\theta)}{\int_\theta L(\theta \mid X, Y)p(\theta)d\theta}$$

A dificuldade consiste em calcular a integral, já que muitas vezes θ é um vetor de parâmetros. De fato, com a exceção de poucos problemas, a integral não tem solução analítica, sendo necessário, mais uma vez, o uso de métodos numéricos.

A ideia é gerar amostras aleatórias independentes da distribuição *a posteriori* para cada parâmetro. Diversos algoritmos executam esse tipo de tarefa como Metropolis-Hastings (Hastings, 1970), Amostrador de Gibbs (Geman e Geman, 1984) ou Monte Carlo Híbrido (Duane *et al.*, 1987). Todos eles fazem parte de uma família chamada Monte Carlo via cadeias de Markov (MCMC) em que um processo estocástico cuja distribuição estacionária é aquela que procuramos é gerado, ou seja, simula-se observações da distribuição *a posteriori*.

A implementação dessa abordagem será mais direta do que a frequentista pois os algoritmos já estão implementados, bastando fornecer a verossimilhança e as *prioris*. O pacote LaplacesDemon (Hall, 2012) possui uma vasta gama de algoritmos para gerar as amostras *a posteriori* e analisar os resultados, ao mesmo tempo em que é simples de utilizar. O único ponto que talvez seja confuso é a criação do objeto com os dados que devem estar em uma lista contendo certos campos específicos indicando a variável resposta, a matriz de plane-

jamento e os parâmetros que serão monitorados. Para facilitar ao usuário, o Código 3.12 implementa uma função que recebe os dados em forma de fórmula e a função de ligação desejada e ela retorna um objeto com os campos necessários e uma sugestão para rodar a análise.

```
1  dataLD <- function(formula, link = c('logit', 'probit', 'cloglog',
       'Aranda-Ordaz', 'Weibull', 'Prentice', 'Stukel'), data, subset){
2    formula <- formula(formula)
3    if(missing(data))
4      data <- environment(formula)
5    if(!missing(subset))
6      data <- subset(data, subset)
7    X    <- model.matrix(formula, data)
8    resp <- model.frame(formula, data)[,1]
9    resp <- factor(resp)
10
11   if(nlevels(resp) > 2)
12     warning('Number of levels greater than 2. The first one will be
           regressed against the others.')
13   if(nlevels(resp) < 2)
14     stop('Response has fewer than two levels.')
15   resp <- ifelse(resp == levels(resp)[1], 0, 1)
16   link <- match.arg(link)
17
18   if(link == 'logit'){
19     parm.names <- as.parm.names(list(beta = rep(0, ncol(X))))
20     model <- 'ModelLogit()'
21   }
22   if(link == 'probit'){
23     parm.names <- as.parm.names(list(beta = rep(0, ncol(X))))
24     model <- 'ModelProbit()'
25   }
26   if(link == 'cloglog'){
27     parm.names <- as.parm.names(list(beta = rep(0, ncol(X))))
28     model <- 'ModelCloglog()'
29   }
```

```
30
31    if(link == 'Aranda-Ordaz'){
32      parm.names <- as.parm.names(list(beta = rep(0, ncol(X)), alpha = 0))
33      model <- 'ModelAranda()'
34    }
35    if(link == 'Weibull'){
36      parm.names <- as.parm.names(list(beta = rep(0, ncol(X)), gamma = 0))
37      model <- 'ModelWeibull()'
38    }
39    if(link == 'Prentice'){
40      parm.names <- as.parm.names(list(beta = rep(0, ncol(X)), m = c(0,
          0)))
41      model <- 'ModelPrentice()'
42    }
43    if(link == 'Stukel'){
44      parm.names <- as.parm.names(list(beta = rep(0, ncol(X)), a = c(0,
          0)))
45      model <- 'ModelStukel()'
46    }
47    dataset <- list(J = ncol(X), X = X, mon.names = 'LP', parm.names =
        parm.names, y = resp)
48
49    cat('Demonic dataset is ready!\n\n')
50    print(data.frame(variable = colnames(X), parameter =
        parm.names[1:ncol(X)]), row.names = F)
51    cat('\nLaplace's Demon suggests running the following code:\n\n')
52    cat('fit = LaplacesDemon(', model, ', data, GIV(', model, ',
        data))\n\n', sep = '')
53    cat('Remember to substitute 'data' with the object returned by this
        function!')
54
55    invisible(dataset)
56  }
```

Código 3.12: *Função para criar objeto com os dados para análise Bayesiana*

É importante notar que são feitas as mesmas verificações da abordagem frequentista.

Outro ponto é que os coeficientes de regressão não mostrarão o nome da variável a que eles se referem nos resultados, então uma tabela com as indicações é mostrada. Caso o usuário esqueça quais são essas atribuições, rodar a função novamente sem atribuir a nenhum objeto mostrará apenas o texto que contém a tabela. Também é necessário salientar que na sugestão de comando é preciso substituir as duas referências a "data" pelo objeto retornado pela função, mas isso também é informado.

Para fazer os ajustes resta apenas programar as funções que calculam a verossimilhança e as distribuições *a priori*. Novamente, apenas o exemplo do modelo de Prentice será mostrado, mas os demais são análogos (Código 3.13).

```
1  ModelPrentice <- function(parm, data, beta.mean = 0, beta.var = 1000,
       m.scale = 1e-10){
2    Model <- function(parm, data){
3      ###parameter
4      beta <- parm[1:data$J]
5      m <- parm[c(data$J + 1, data$J + 2)]
6      ###priors
7      beta.prior <- dnormv(beta, beta.mean, beta.var, log = TRUE)
8      m.prior <- dhalfnorm(m, m.scale, log = TRUE)
9      ###logLik
10     LL <- -likprentice(m, beta, data$y, data$X)
11     ###log-posterior
12     LP <- LL + sum(beta.prior) + sum(m.prior)
13     Modelout <- list(LP = LP, Dev = -2*LL, Monitor = LP, mu = data$X
           %*% beta, parm = parm)
14     return(Modelout)
15   }
16   return(Model)
17 }
```

Código 3.13: *Modelo Bayesiano para ligação de Prentice*

Segundo a documentação do pacote LaplacesDemon, a função com o modelo deve ter dois parâmetros exatamente e a especificação das prioris deve ser feita manualmente. Por esse motivo, o algoritmo apresentado retorna outra função com essas restrições, mas permite

alterar os hiperparâmetros de forma conveniente sem precisar escrever uma nova função. É claro que se o usuário quiser modificar a família das distribuições *a priori*, terá que fazer isso manualmente. Outra observação sobre os hiperparâmetros é que os valores fornecidos serão reciclados caso não estejam em número suficiente. Isto quer dizer que se há cinco coeficientes de regressão para estimar e apenas três médias (e variâncias) para as *prioris* são fornecidas, então os dois primeiros valores serão usados para o quarto e quinto parâmetros.

Uma amostra *a posteriori* será simulada com uma chamada a `LaplacesDemon()` em que deve ser informado o modelo a ser utilizado, a lista com os dados, algoritmo a ser utilizado, o número de iterações e tamanho do salto. Este último valor serve para eliminar a autocorrelação da amostra gerada. O número de iterações iniciais descartadas para garantir estacionariedade (*burnin*) é determinado automaticamente utilizando as recomendações em Geweke (1992). Avaliação dos resultados pode ser feita com chamadas a `plot()` contendo a amostra gerada e o modelo utilizado ou sugestão de melhorias são fornecidas com o comando `Consort()`. Exemplos de utilização desses procedimentos serão apresentados no próximo capítulo.

Capítulo 4

Exemplos de Aplicação

Neste capítulo dois bancos de dados serão analisados com intuito de mostrar como é a utilização das rotinas e comparar alguns resultados. Primeiro será estudado o banco de dados sobre mortalidade de besouros (Bliss, 1935) e um segundo será sobre a presença de micronúcleos em células submetidas a doses de radiação (Balasem e Ali, 1991). As abordagens clássica e Bayesiana serão utilizadas, bem como todas as funções descritas no Capítulo 3, destacando os comandos.

4.1 Mortalidade de besouros

O objetivo original desse estudo era obter um inseticida eficaz contra besouros. Para isso, 481 animais foram expostos a diferentes concentrações de dissulfeto de carbono (CS_2) durante cinco horas e contou-se o número de insetos mortos. Esse conjunto de dados é conhecido por não ser bem ajustados por modelos simétricos, em particular logito e probito; por conta disso, é amplamente citado em trabalhos que buscam alternativas a esses modelos.

4.1.1 Abordagem frequentista

A Tabela 4.1 mostra a proporção de besouros mortos em cada faixa de concentração do inseticida bem como as proporções estimadas de cada um dos modelos estudados. É sabido que o modelo complementar log-log se ajusta muito bem a esses dados, mas como ele pode ser aproximado pelos paramétricos apresentados, percebe-se que todos mostram estimativas

muito próximas ao observado. Os modelos logito e probito, no entanto, ajustam bem apenas as probabilidades de uma das caudas, evidenciando a natureza assimétrica do problema.

	ln(Dose)							
Modelo	1,6907	1,7242	1,7552	1,7842	1,8113	1,8369	1,861	1,8839
Observado	0,102	0,217	0,290	0,500	0,825	0,898	0,984	1,000
Logito	0,059	0,164	0,362	0,605	0,795	0,903	0,955	0,979
Probito	0,057	0,179	0,379	0,604	0,788	0,904	0,962	0,987
Complementar log-log	0,095	0,188	0,338	0,542	0,758	0,918	0,986	0,999
Aranda-Ordaz	0,095	0,188	0,338	0,542	0,758	0,918	0,986	0,999
Weibull	0,090	0,186	0,340	0,548	0,762	0,917	0,984	0,999
Prentice	0,101	0,187	0,327	0,533	0,767	0,927	0,984	0,997
Stukel	0,118	0,184	0,304	0,524	0,780	0,930	0,983	0,996

Tabela 4.1: *Proporções observadas e estimadas pela abordagem frequentista de besouros mortos por nível de dose*

O banco de dados contém 481 linhas, cada uma com a dose de inseticida (no caso, é utilizado seu logaritmo) e se o animal morreu após a exposição. A sequência de comandos seguinte ajusta todos esses modelos na abordagem frequentista:

```
f1 = glm(mortos ~ ldose, binomial)
f2 = glm(mortos ~ ldose, binomial("probit"))
f3 = glm(mortos ~ ldose, binomial("cloglog"))
f4 = binreg(mortos ~ ldose, "A")
f5 = binreg(mortos ~ ldose, "W")
f6 = binreg(mortos ~ ldose, "P")
f7 = binreg(mortos ~ ldose, "S")
```

Note como não é necessário escrever o nome inteiro da ligação, já que a primeira letra de cada uma é diferente. A comparação de alguns critérios de informação pode ser obtida com a chamada `compareIC(f1, f2, f3, f4, f5, f6, f7)`. A saída deste comando é a Tabela 4.2 exceto a coluna do tempo de execução. Por ela percebe-se que o modelo Stukel tem a maior verossimilhança, mas os critérios de informação são menores para o complementar log-log, indicando que a melhoria obtida pelos parâmetros extras não é grande o suficiente.

Uma outra ferramenta que o código trás é a comparação dos modelos paramétricos com o logístico através do teste da razão de verossimilhanças. Executar o comando `LRTlogit(f7)`

compara o modelo Stukel com o logístico resultando em $\chi^2(2) = 8{,}489$, $p = 0{,}014$, ou seja, o modelo Stukel ajusta os dados significativamente melhor do que o logito. Fazendo isso com os outros modelos paramétricos sempre resulta em melhora do ajuste com relação à regressão logística (Aranda-Ordaz: $\chi^2(1) = 7{,}786$, $p = 0{,}005$; Weibull: $\chi^2(1) = 7{,}670$, $p = 0{,}005$; Prentice: $\chi^2(2) = 7{,}971$, $p = 0{,}019$).

Modelo	g.l.	log-verossimilhança	AIC	BIC	Tempo de Execução (s)
Logito	2	-186,23	376,47	384,82	0,027
Probito	2	-185,68	375,36	383,71	0,025
Complementar log-log	2	-182,34	368,68	377,04	0,025
Aranda-Ordaz	3	-182,34	370,68	383,23	0,400
Weibull	3	-182,41	370,81	383,34	6,448
Prentice	4	-182,25	372,50	389,20	2,627
Stukel	4	-181,99	371,98	388,68	2,692

Tabela 4.2: *Comparação dos modelos ajustados pelo abordagem frequentista para dados de mortalidade de besouros*

Os coeficientes de regressão, parâmetros de ligação podem ser obtidos simplesmente digitando o nome do modelo; maiores informações como erro-padrão e teste de significância podem ser obtidos com a função `summary`. Um resumo desses dados estão na Tabela 4.3. Interessante notar que apesar dos modelos de ligação paramétrica não possuírem o melhor ajuste, eles tendem a se aproximar do complementar log-log através dos parâmetros de ligação. Em particular o modelo Aranda-Ordaz tem $\alpha = 2{,}817 \times 10^{-6}$ que é muito próximo de zero, valor que faria esse modelo ser equivalente ao ótimo. O Weibull também tem essa característica com um valor alto de γ.

As Figuras 4.1 e 4.2 mostram como cada função de ligação se ajusta aos dados observados. Novamente é possível ver como as curvas logito e probito não passam muito próximas aos pontos, mas as demais estão praticamente em cima dos dados observados.

	Coef.	EP	z	Valor-p
Logito				
Intercepto	-60,717	5,181	-11,720	$< 0{,}001$
ldose	34,270	2,912	11,770	$< 0{,}001$
Probito				
Intercepto	-34,935	2,648	-13,190	$< 0{,}001$
ldose	19,728	1,487	13,270	$< 0{,}001$
Complementar log-log				
Intercepto	-39,572	3,240	-12,210	$< 0{,}001$
ldose	22,041	1,799	12,250	$< 0{,}001$
Aranda-Ordaz				
Intercepto	-39,527	3,240	-12,200	$< 0{,}001$
ldose	22,041	1,799	12,200	$< 0{,}001$
α	$1{,}180 \times 10^{-6}$	0,074	-13,435	$< 0{,}001$
Weibull				
Intercepto	-0,129	0,091	-1,411	0,158
ldose	0,629	0,051	12,421	$< 0{,}001$
γ	34,861	2,808	11,134	$< 0{,}001$
Prentice				
Intercepto	-91,670	8,134	-11,270	$< 0{,}001$
ldose	50,207	4,521	11,110	$< 0{,}001$
m_1	0,366	0,041	-15,280	$< 0{,}001$
m_2	1,531	0,258	2,060	0,039
Stukel				
Intercepto	-69,534	7,123	-9,762	$< 0{,}001$
ldose	39,025	3,956	9,865	$< 0{,}001$
a_1	0,164	0,092	1,790	0,074
a_2	-0,521	0,219	-2,380	0,017

Tabela 4.3: *Coeficientes de regressão ajustados pela abordagem frequentista para os dados de mortalidade de besouros*

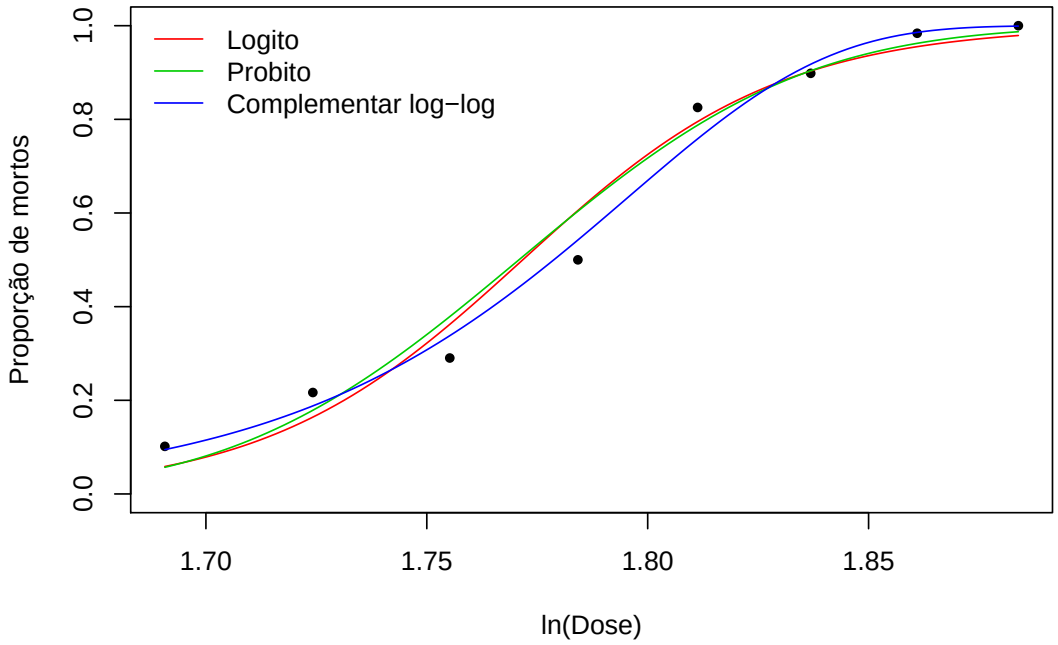

Figura 4.1: *Ajuste dos modelos de ligação não paramétrica para dados de mortalidade de besouros*

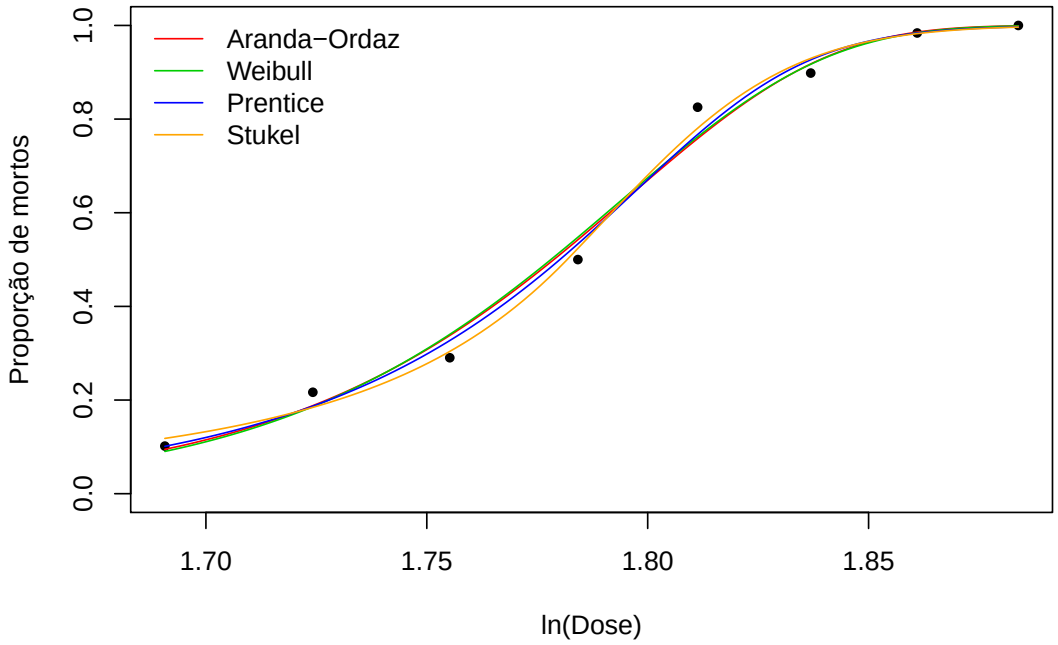

Figura 4.2: *Ajuste dos modelos de ligação paramétrica para dados de mortalidade de besouros*

4.1.2 Abordagem Bayesiana

Fazer análise Bayesiana exige um pouco mais de cuidado com o problema. A escolha das distribuições *a priori* pode diferir daquelas apresentadas no código, então o próprio usuário deverá modificá-la. É aconselhável que copie código do modelo que se deseja utilizar e crie uma nova função alterando as linhas correspondentes à *priori*. Para efeito deste exemplo será utilizado os valores padrão correspondentes a *prioris* não informativas.

Um segundo cuidado a ser tomado é qual algoritmo utilizar. Por padrão é rodado o Metropolis-Hastings com passeio aleatório. A distribuição proposta para a amostragem é calculada automaticamente e elas podem se adaptar em intervalos fornecido pelo usuário. Este último método é conhecido como Metropolis adaptativo (Haario *et al.*, 2001) que se mostrou muito útil para ajustar os modelos de regressão binária.

O primeiro passo da análise é criar o objeto contendo a lista dos dados e parâmetros que serão utilizados na análise. Isso é feito com a função `dataLD` que recebe como parâmetro a fórmula da equação, a função de ligação, o objeto com os dados e eventuais restrições. Essa função retorna a lista e imprime uma proposta inicial de código a ser rodado. Essa sugestão utiliza o algoritmo Metropolis-Hastings por 100.000 iterações, retendo uma amostra a cada 100 passos e *prioris* não informativas. Um exemplo para o modelo complementar log-log é o seguinte:

```
dados = dataLD(mortos ~ ldose, "cloglog")
fit = LaplacesDemon(ModelCloglog(), dados, GIV(ModelCloglog(), dados))
```

Se o usuário desejar modificar os hiperparâmetros, basta incluir dentro da chamada de `ModelCloglog()`. Por exemplo, para especificar as *prioris* Normais com médias -50 e 30 e variâncias 10 e 10 para o intercepto e ldose, respectivamente, basta fazer:

```
fit = LaplacesDemon(ModelCloglog(beta.mean = c(-50, 30), beta.var = 10),
                                                dados, GIV(ModelCloglog(), dados))
```

A Tabela 4.4 resume os resultados dos modelos com critério de informação do desvio (DIC, Spiegelhalter *et al.*, 2002), tempo de execução, número de iterações necessárias para obter estacionariedade e qual algoritmo foi utilizado. O objetivo deste exemplo é chegar a

1000 amostras para cada parâmetro, mas devido ao tamanho dos saltos para evitar a dependência e ao período de *burnin*, diferentes números totais de iterações foram necessários. A qualidade da amostragem foi medida basicamente com o erro padrão do Monte Carlo e tamanho amostral efetivo que compara dois métodos de amostragem. Além disso, o algoritmo deveria apresentar uma taxa de aceitação razoável; Roberts *et al.* (1997) recomenda taxas de aproximadamente 50% para um estimador e este número pode baixar para aproximadamente 23% conforme a quantidade de parâmetros aumenta. Felizmente essas checagens podem ser facilmente obtidas com o comando `Consort(fit)` em que amostra será analisada e caso as condições não sejam favoráveis, um novo comando será sugerido para melhorar os resultados.

Modelo	DIC	Tempo de execução (min)	Iterações	Algoritmo
Logito	376,42	0,57	10^5	Metropolis-Hastings
Probito	375,56	1,33	$1{,}3 \times 10^5$	Metropolis adaptativo
Complementar log-log	368,65	0,28	$1{,}3 \times 10^5$	Metropolis adaptativo
Aranda-Ordaz	370,92	11,7	$1{,}4 \times 10^6$	Metropolis-Hastings
Weibull	370,87	8,30	$1{,}1 \times 10^6$	Metropolis adaptativo
Prentice	371,92	48,51	$8{,}3 \times 10^5$	Metropolis adaptativo robusto
Stukel	374,45	12,19	$3{,}1 \times 10^5$	Metropolis adaptativo

Tabela 4.4: *Descrição dos modelos Bayesianos ajustados para os dados de mortalidade de besouros*

Novamente o modelo complementar log-log obteve o melhor desempenho para ajustar os dados e também é o algoritmo que tem o menor tempo de execução (16,8 segundos), o que é bastante rápido para análises Bayesianas. Dentre os modelos paramétricos, Aranda-Ordaz e Weibull têm praticamente o mesmo DIC como ocorreu com o AIC na abordagem frequentista.

Essas informações são apresentadas simplesmente digitando o nome do modelo no R. Além disso, estatísticas dos parâmetros são apresentadas para toda a simulação e também apenas para a parte estacionária. A Tabela 4.5 mostra as médias dos parâmetros que é o estimador pontual Bayesiano assumindo função de perda quadrática (DeGroot, 2004) e seus desvios-padrão. O intervalo de credibilidade 95% é baseado nos quantis amostrais de 2,5% e 97,5% e serve como uma alternativa aos testes de significância.

Em geral as estimativas são bem próximas ao equivalente frequentista devido às *prioris* não informativas. Uma diferença notável é o modelo Prentice que apresenta o parâmetro de

ligação m_2 muito alto, mas isso é a maneira desse modelo para se aproximar do complementar log-log como notado na Seção 2.1.6.

			Intervalo de Credibilidade 95%	
	Média	DP	Lim. Inferior	Lim. Superior
Logito				
Intercepto	-59,25	5,93	-71,76	-49,80
ldose	33,45	3,34	28,05	40,46
Probito				
Intercepto	-28,27	0,25	-28,71	-27,83
ldose	15,98	0,15	15,74	16,24
Complementar log-log				
Intercepto	-39,33	3,16	-45,59	-33,40
ldose	21,90	1,75	18,60	25,35
Aranda-Ordaz				
Intercepto	-44,72	5,75	-57,63	-35,50
ldose	25,02	3,28	19,85	32,37
α	0,26	0,22	0,01	0,81
Weibull				
Intercepto	-2,59	2,52	-8,63	0,63
ldose	2,00	1,41	0,20	5,38
γ	22,54	25,02	3,86	107,73
Prentice				
Intercepto	-40,91	12,19	-73,19	-26,93
ldose	14,45	6,39	7,17	31,55
m_1	3,10	1,92	0,55	7,04
m_2	$8{,}60 \times 10^6$	$1{,}66 \times 10^6$	$5{,}49 \times 10^6$	$1{,}15 \times 10^7$
Stukel				
Intercepto	-52,39	14,40	-82,78	-26,91
ldose	29,42	8,06	15,12	46,47
a_1	0,57	0,48	-0,01	1,88
a_2	-0,23	0,48	-1,17	0,74

Tabela 4.5: *Estatísticas da amostra* a posteriori *estacionária para os dados de mortalidade de besouros*

Gráficos contendo o movimento da cadeia gerada, estimativa da função densidade e gráfico da função de autocorrelação podem ser obtidos para cada parâmetro com o comando `plot(fit, fit$Rec.BurnIn.Thinned, dados)`. Outras quantidades monitoradas como a função desvio e o valor do logaritmo da *posteriori* também serão apresentadas. A Figura 4.3 apresenta a saída deste comando para o modelo Aranda-Ordaz. Os parâmetros `beta[1]` e `beta[2]` são o intercepto e o coeficiente para o logaritmo da dose, respectivamente. A linha vermelha no gráfico do histórico das iterações mostra a evolução da média. As densidades estimadas mostram-se unimodais, mas ligeiramente assimétricas. A função de autocorrelação

também apresenta praticamente nenhuma correlação serial, indicando que a escolha do tamanho dos saltos foi adequada.

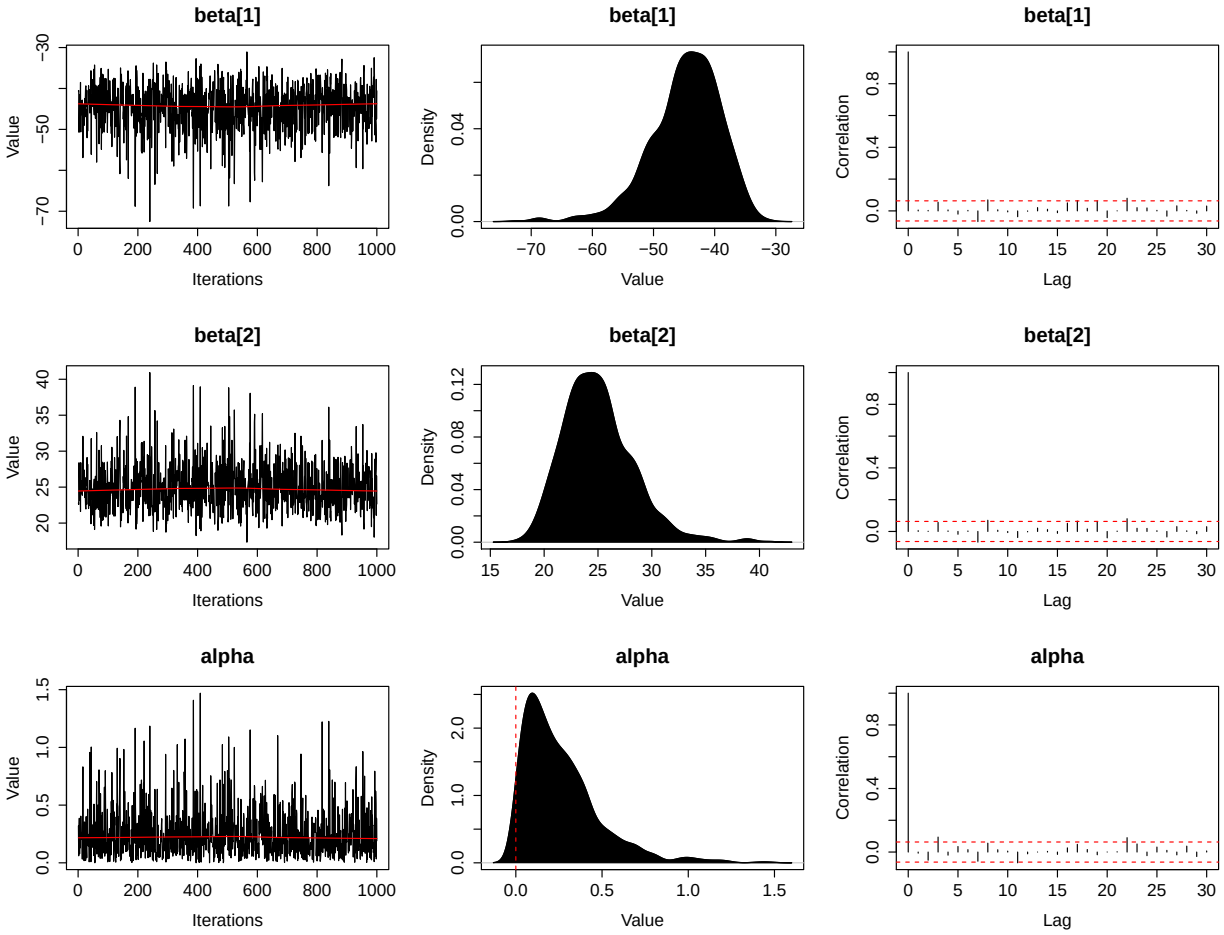

Figura 4.3: *Amostras* a posteriori *dos parâmetros do modelo Aranda-Ordaz para os dados de mortalidade de besouros*

4.2 Frequência de micronúcleos

Micronúcleo é o nome dado a um pequeno núcleo formado sempre que um cromossomo ou fragmento de cromossomo não é incorporado ao núcleo das células filhas durante o processo de divisão celular. Sua presença é um índice de toxicidade genética. Em Balasem e Ali (1991) é estudado o efeito de 10 diferentes doses de radiação por raios gama de Césio 137 em linfócitos e avalia a frequência de micronúcleos em 5007 células. No estudo original é proposto um modelo quadrático para contar o número de micronúcleos, porém estes dados serão utilizados para demonstrar que os modelos de ligação paramétrica se ajustam melhor para calcular a proporção de células com micronúcleos.

4.2.1 Abordagem frequentista

A Tabela 4.6 mostra as proporções observadas e estimadas de células com micronúcleos por dose de radiação. Os modelos com ligação não paramétrica têm dificuldade em estimar as probabilidades muito baixas dos três primeiros níveis de dose de radiação, enquanto que os paramétricos não apresentam essa dificuldade. Para as outras doses não há muita diferença nas estimativas, mas o modelo Stukel se destaca por passar em cima de praticamente todos os pontos como mostra a Figura 4.5. Na Figura 4.4 é possível ver como os modelos usuais não têm um bom desempenho em boa parte das estimativas.

	Dose de radiação (cGy)									
Modelo	5	10	25	50	100	200	300	400	500	600
Observado	0,038	0,046	0,058	0,100	0,138	0,322	0,392	0,521	0,652	0,759
Logito	0,074	0,077	0,084	0,097	0,130	0,223	0,356	0,515	0,671	0,797
Probito	0,068	0,070	0,078	0,094	0,130	0,231	0,364	0,516	0,666	0,793
Complementar log-log	0,088	0,090	0,097	0,109	0,137	0,214	0,325	0,473	0,649	0,819
Aranda-Ordaz	0,046	0,049	0,060	0,082	0,140	0,290	0,435	0,555	0,650	0,725
Weibull	0,041	0,046	0,061	0,088	0,149	0,283	0,419	0,544	0,653	0,742
Prentice	0,046	0,049	0,061	0,084	0,141	0,283	0,428	0,553	0,653	0,732
Stukel	0,044	0,048	0,060	0,084	0,146	0,296	0,430	0,527	0,635	0,763

Tabela 4.6: *Proporções observadas e estimadas pela abordagem frequentista de presença de micronúcleos por nível de radiação*

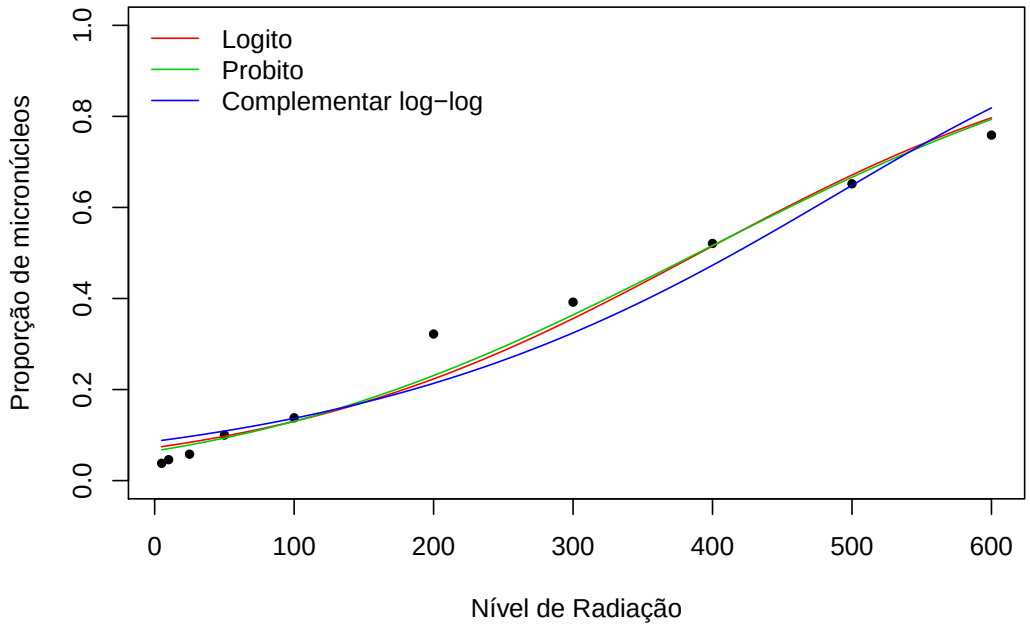

Figura 4.4: *Ajuste dos modelos de ligação não paramétrica para dados de frequência de micronúcleos*

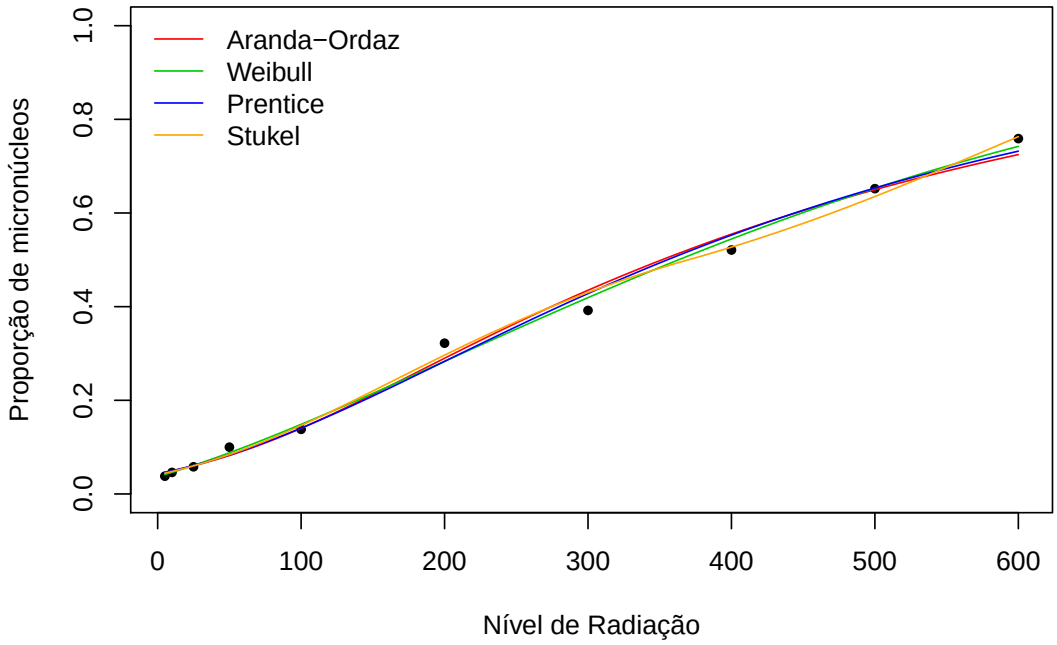

Figura 4.5: *Ajuste dos modelos de ligação paramétrica para dados de frequência de micronúcleos*

Ajustar os modelos é análogo ao exemplo anterior, bastando substituir a fórmula para as variáveis corretas:

```
f1 = glm(mn ~ dose, binomial)
f2 = glm(mn ~ dose, binomial("probit"))
f3 = glm(mn ~ dose, binomial("cloglog"))
f4 = binreg(mn ~ dose, "A")
f5 = binreg(mn ~ dose, "W")
f6 = binreg(mn ~ dose, "P")
f7 = binreg(mn ~ dose, "S")
```

Executar o comando `LRTlogit` em todos os modelos confirma que eles são significativamente melhores do que o logístico em termos de poder de predição (Aranda-Ordaz: $\chi^2(1) = 43{,}602$, $p < 0{,}001$; Weibull: $\chi^2(1) = 49{,}646$, $p < 0{,}001$; Prentice: $\chi^2(2) = 45{,}287$, $p < 0{,}001$; Stukel: $\chi^2(2) = 50{,}549$, $p < 0{,}001$). Além disso, os critérios de informação apontam os modelos Weibull e Stukel como os melhores (Tabela 4.7, gerada com o comando `compareIC`), mas são praticamente indistinguíveis entre si em qualidade de ajuste.

Modelo	g.l.	log-verossimilhança	AIC	BIC	Tempo de execução (s)
Logito	2	-2275,48	4554,95	4567,99	0,137
Probito	2	-2268,48	4540,97	4554,01	0,117
Complementar log-log	2	-2297,33	4598,66	4611,70	0,119
Aranda-Ordaz	3	-2253,68	4513,35	4532,91	13,103
Weibull	3	-2250,65	4507,31	4526,86	5,503
Prentice	4	-2252,83	4513,67	4539,74	156,688
Stukel	4	-2250,20	4508,40	4534,48	247,138

Tabela 4.7: *Comparação dos modelos ajustados pela abordagem frequentista para dados de frequência de micronúcleos*

	Coef.	EP	z	Valor-p
Logito				
Intercepto	-2,5524	0,0673	-37,9355	$< 0,001$
dose	0,0065	0,0002	34,4396	$< 0,001$
Probito				
Intercepto	-1,5136	0,0355	-42,6036	$< 0,001$
dose	0,0039	0,0001	37,1818	$< 0,001$
Complementar log-log				
Intercepto	-2,4060	0,0575	-41,8427	$< 0,001$
dose	0,0049	0,0001	36,1386	$< 0,001$
Aranda-Ordaz				
Intercepto	-2,9719	0,1066	-27,8700	$< 0,001$
dose	0,0164	0,0005	31,6864	$< 0,001$
α	6,8253	0,2368	24,6003	$< 0,001$
Weibull				
Intercepto	0,1148	0,0093	12,3007	$< 0,001$
dose	0,0018	0,0001	38,0708	$< 0,001$
γ	1,5153	0,0399	-52,2956	$< 0,001$
Prentice				
Intercepto	0,6421	0,0616	10,4288	$< 0,001$
dose	0,0098	0,0003	33,2890	$< 0,001$
m_1	3,5921	0,2173	11,9287	$< 0,001$
m_2	0,2643	0,0106	-69,2532	$< 0,001$
Stukel				
Intercepto	-1,2709	0,0197	-64,5208	$< 0,001$
dose	0,0034	0,0001	44,3080	$< 0,001$
a_1	0,9338	0,1927	4,8451	$< 0,001$
a_2	1,2661	0,0436	29,0477	$< 0,001$

Tabela 4.8: *Coeficientes de regressão ajustados pela abordagem frequentista para os dados de frequência de micronúcleos*

Um ponto negativo nesse exemplo é o tempo de execução elevado, chegando a quase oito minutos para o modelo Stukel. O modelo Weibull leva grande vantagem nesse caso, sendo o mais rápido entre os de ligação paramétrica. Os coeficientes de regressão podem ser obtidos com a função `summary` e estão na Tabela 4.8. Importante notar como todos os parâmetros são altamente significativos, em especial os de ligação. Isto é mais um indício da melhor adequabilidade desses modelos com relação ao logito. Outra observação é que os parâmetros de ligação não ficam perto de nenhum caso especial descrito no Capítulo 2, o que mostra como esses modelos são capazes de tratar casos singulares como o deste exemplo.

4.2.2 Abordagem Bayesiana

Novamente a distribuições *a priori* serão não informativas para ajustar as regressões. Uma característica desse banco de dados foi a dificuldade em obter a estacionariedade das distribuições com métodos não adaptativos, por isso o Metropolis adaptativo foi utilizado em todos os casos. Portanto, a sugestão dada pela função `dataLD` não é a mais adequada para esse caso, mas o seguinte código resolve o problema para o caso Weibull e é análogo para os outros modelos:

```
dados = dataLD(mn ~ dose, "Weibull")
fit = LaplacesDemon(ModelWeibull(), dados, GIV(ModelWeibull(), dados),
         Algorithm = "AM", Specs = list(Adaptive = 600, Periodicity = 4000))
```

Esse comando também executa 100.000 iterações com salto de tamanho 100*. Em `Specs` é detalhado que nas primeiras 600 iterações não ocorre nenhuma adaptação e depois disso, a matriz de covariância da distribuição proposta é recalculada a cada 4000 passos. Diminuir a periodicidade ajuda na convergência do algoritmo ao mesmo tempo em que deixa-o mais lento. Novamente é recomendado rodar uma amostra inicial e avaliar o resultado com `Consort` para que o programa decida a periodicidade mais adequada para o problema.

A Tabela 4.9 resume alguns resultados como critério de informação e tempo de execução. Novamente há uma tendência das ligações Weibull e Stukel se ajustarem melhor do que as

*Argumentos *Iterations* e *Thinning*, respectivamente, omitidos pois seus valores padrão estão sendo utilizados.

demais, mas sendo praticamente iguais entre si. O tempo de execução elevado deve-se tanto ao método adaptativo que faz contas a mais quanto ao tamanho da amostra dez vezes maior que o anterior, mas a diferença para o outro banco de dados não é tão grande assim.

Modelo	DIC	Tempo de execução (min)	Iterações	Algoritmo
Logito	4554,13	4,22	$1{,}3 \times 10^5$	Metropolis adaptativo
Probito	4540,19	8,71	$1{,}3 \times 10^5$	Metropolis adaptativo
Complementar log-log	4597,79	7,17	$1{,}3 \times 10^5$	Metropolis adaptativo
Aranda-Ordaz	4520,54	4,64	$1{,}3 \times 10^5$	Metropolis adaptativo
Weibull	4507,54	29,02	10^5	Metropolis adaptativo
Prentice	4510,62	4,39	$1{,}3 \times 10^5$	Metropolis adaptativo
Stukel	4507,28	24,97	$1{,}3 \times 10^5$	Metropolis adaptativo

Tabela 4.9: *Descrição dos modelos Bayesianos ajustados para os dados de frequência de micronúcleos*

Estatísticas das amostras geradas podem ser encontradas na Tabela 4.10 e, mais uma vez, os resultados são semelhantes aos encontrados na análise frequentista. Nenhum dos intervalos de credibilidade contêm valores críticos para os parâmetros serem desconsiderados num eventual teste de hipóteses assim como ocorreu na outra abordagem. A Figura 4.6 traz resultados específicos para o modelo Weibull com gráficos de histórico de iterações, densidade estimada e função de autocorrelação para os três parâmetros. As densidades do intercepto (`beta[1]`) e para γ mostram indícios de bimodalidade, mas os picos da distribuição estão perto o suficiente para não causar maiores problemas na estimativas pela média.

			Intervalo de Credibilidade 95%	
	Média	DP	Lim. Inferior	Lim. Superior
Logito				
Intercepto	-2.5567	0.0680	-2.7028	-2.4325
dose	0.0065	0.0002	0.0062	0.0069
Probito				
Intercepto	-1.5139	0.0322	-1.5817	-1.4501
dose	0.0039	0.0001	0.0037	0.0041
Complementar log-log				
Intercepto	-2.4067	0.0426	-2.5335	-2.3341
dose	0.0049	0.0001	0.0048	0.0052
Aranda-Ordaz				
Intercepto	-3.0516	0.1412	-3.3675	-2.7826
dose	0.0216	0.0069	0.0136	0.0393
α	9.6862	3.7902	5.2122	19.9662
Weibull				
Intercepto	0.1256	0.0371	0.0638	0.2106
dose	0.0018	0.0001	0.0016	0.0020
γ	1.5613	0.1637	1.2842	1.9200
Prentice				
Intercepto	2.2323	0.5525	1.1359	3.3744
dose	0.0100	0.0038	0.0058	0.0205
m_1	16.1856	7.7203	4.2407	35.8024
m_2	0.2899	0.1136	0.0991	0.5180
Stukel				
Intercepto	-1.2597	0.1429	-1.5304	-1.0146
dose	0.0034	0.0004	0.0028	0.0041
a_1	0.9486	0.5006	0.0543	1.8834
a_2	1.3262	0.3331	0.7898	1.9882

Tabela 4.10: *Estatísticas da amostra* a posteriori *estacionária para os dados de mortalidade de besouros*

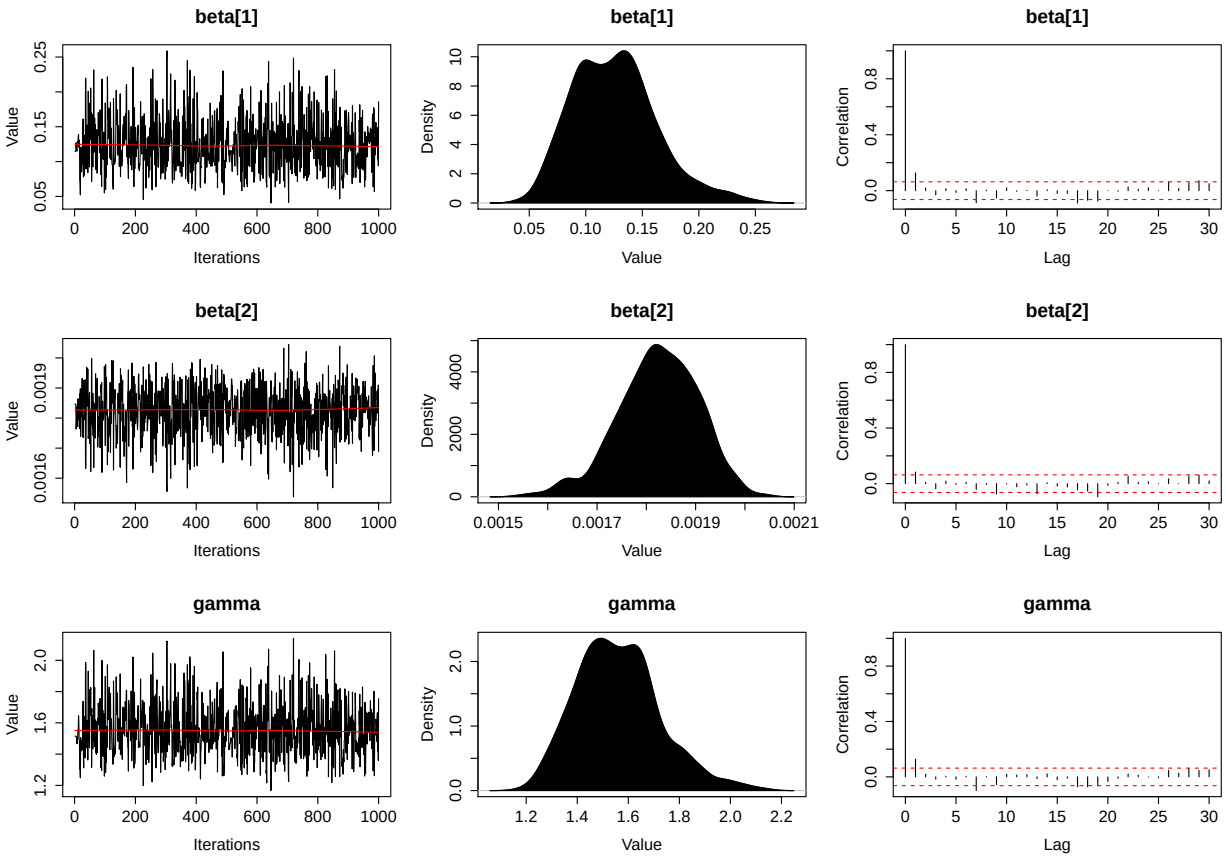

Figura 4.6: *Amostras* a posteriori *dos parâmetros do modelo Weibull para os dados de frequência de micronúcleos*

Capítulo 5

Considerações Finais

Este trabalho apresentou um compêndio de códigos em R para realizar análise de modelos de regressão binária com função de ligação paramétrica. Diversas propostas já haviam sido formuladas para generalizar o modelo logito, porém não havia nenhum pacote estatístico programado para ajustá-los. As rotinas escritas são de código aberto, significando que é de livre acesso para usuários modificarem-na. Métodos de estimação foram implementados para as inferências Frequentista e Bayesiana e os códigos permitem incorporar novos modelos facilmente devido à abordagem geral utilizada.

Exemplos de aplicação mostraram que as funções são simples de usar permitindo que diversos comandos familiares ao usuário de R sejam utilizados diretamente. Além disso, rotinas extras facilitam o trabalho de comparação de modelos seja por critérios de informação, função de Verossimilhança ou teste de hipóteses.

Dos bancos de dados estudados, um é comumente apresentado quando se trata de regressão binária e possui uma solução bastante satisfatória com o modelo complementar log-log. No entanto, as regressões com função de ligação paramétrica tenderam a aproximá-lo já que esse é um caso particular para os quatro modelos apresentados. Um segundo banco de dados mostra a ineficiência dos modelos usuais na qualidade de ajuste, mas todos os de ligação paramétrica possuem soluções bastante satisfatórias e são superiores em todos os critérios de informação.

Os códigos, apesar de funcionais, podem ser otimizados, principalmente os referentes à abordagem Frequentista. Há um *bug* conhecido ainda sem solução relacionado ao argumento

`subset` dos comandos `binreg` e `dataLD`. O objeto passado nesse campo não herda ambiente da fórmula. Digamos que os dados a ser analisados estão em um `data.frame` de nome `dados` e o usuário deseja selecionar apenas as observações que obedecem certas condições envolvendo a variável `x` dentro de `dados`. Isso, então, deve ser passado usando `dados$x` ao invés de apenas o nome da variável como em outras funções do R.

Apesar de ter sido testado em alguns bancos de dados com sucesso, a quantidade não é suficiente para garantir a ausência de problemas no código. Porém, essas rotinas produzem um bom avanço na quantidade de modelos de regressão binária implementados disponíveis para o usuário.

Apêndice A

Rotinas

A.1 Funções Auxiliares

```
pprentice <- function(q, m = c(1, 1)) {
  if(length(m) < 2)
    stop("m = ", deparse(substitute(m)), " should have exactly two
      elements.")
  if(length(m) > 2)
    warning("Length of m = ", deparse(substitute(m)), "is greater than 2.
      Only the first two elements will be used.")

  return( pbeta(plogis(q), m[1], m[2]) )
}

qprentice <- function(p, m = c(1, 1)) {
  if(length(m) < 2)
    stop("m = ", deparse(substitute(m)), " should have exactly two
      elements.")
  if(length(m) > 2)
    warning("Length of m = ", deparse(substitute(m)), "is greater than 2.
      Only the first two elements will be used.")

  return( qlogis(qbeta(p, m[1], m[2])) )
}

dprentice <- function(x, m = c(1, 1)) {
  if(length(m) < 2)
    stop("m = ", deparse(substitute(m)), " should have exactly two
      elements.")
```

```
  if(length(m) > 2)
    warning("Length of m = ", deparse(substitute(m)), "is greater than 2.
      Only the first two elements will be used.")

  exp(x*m[1])*((1 + exp(x))^(-(m[1] + m[2])))/beta(m[1], m[2])
}


hstukel <- function(eta, alpha){
  if(length(alpha) < 2)
    stop("alpha = ", deparse(substitute(alpha)), " should have exactly two
      elements.")
  if(length(alpha) > 2)
    warning("Length of alpha = ", deparse(substitute(alpha)), "is greater
      than 2. Only the first two elements will be used.")

  h <- function(eta){
    if(eta > 0){
      if(alpha[1] > 0) out <- (exp(alpha[1]*eta) - 1)/alpha[1]
      else if(alpha[1] == 0) out <- eta
      else out <- -log(1 - alpha[1]*eta)/alpha[1]
    } else {
      if(alpha[2] > 0) out <- -(exp(-alpha[2]*eta) - 1)/alpha[2]
      else if(alpha[2] == 0) out <- eta
      else out <- log(1 + alpha[2]*eta)/alpha[2]
    }
    return(out)
  }

  return(sapply(eta, h))
}


dhstukel <- function(eta, alpha){
  if(length(alpha) < 2)
    stop("alpha = ", deparse(substitute(alpha)), " should have exactly two
      elements.")
  if(length(alpha) > 2)
    warning("Length of alpha = ", deparse(substitute(alpha)), "is greater
      than 2. Only the first two elements will be used")
```

```
  h <- function(eta){
    if(eta > 0){
      if(alpha[1] > 0) out <- exp(alpha[1]*eta)
      else if(alpha[1] == 0) out <- 1
      else out <- 1/(1 - alpha[1]*eta)
    } else {
      if(alpha[2] > 0) out <- exp(-alpha[2]*eta)
      else if(alpha[2] == 0) out <- 1
      else out <- 1/(1 + alpha[2]*eta)
    }
    return(out)
  }

  return(sapply(eta, h))
}


invhstukel <- function(y, alpha){
  if(length(alpha) < 2)
    stop("alpha = ", deparse(substitute(alpha)), " should have exactly two
      elements.")
  if(length(alpha) > 2)
    warning("Length of alpha = ", deparse(substitute(alpha)), "is greater
      than 2. Only the first two elements will be used")

  h <- function(y){
    if(y > 0){
      if(alpha[1] > 0) out <- log(y*alpha[1] + 1)/alpha[1]
      else if(alpha[1] == 0) out <- y
      else out <- -(exp(-y*alpha[1]) - 1)/alpha[1]
    } else {
      if(alpha[2] > 0) out <- -log(1 - y*alpha[2])/alpha[2]
      else if(alpha[2] == 0) out <- y
      else out <- (exp(y*alpha[2]) - 1)/alpha[2]
    }
  }

  return(sapply(y, h))
}
```

A.2 Funções de Verossimilhança

```
likaranda <- function(alpha, beta, y, X) {
  if(alpha > 0){
    eta <- X %*% beta
    aranda <- ifelse(alpha*exp(eta) > -1, 1 - (alpha*exp(eta) + 1)^(-1/alpha), 1)
    aranda <- pmin(pmax(aranda, .Machine$double.eps),
      1 - .Machine$double.eps)
    return(-sum(dbern(y, aranda, log = TRUE)))
  } else{
    return(1e100)
  }
}

likweibull = function(gama, beta, y, X){
  if (gama > 0) {
    eta <- X %*% beta
    eta <- pmax(eta, 0)
    weibull <- 1 - exp(-eta^gama)
    weibull <- pmin(pmax(weibull, .Machine$double.eps),
      1 - .Machine$double.eps)
    return(-sum(dbern(y, weibull, log = TRUE)))
  } else {
    return(1e100)
  }
}

likprentice <- function(m = c(1, 1), beta, y, X) {
  if(all(m > 0)){
    eta <- X %*% beta
    prentice <- pprentice(eta, m)
    prentice <- pmin(pmax(prentice, .Machine$double.eps),
      1 - .Machine$double.eps)
    return(-sum(dbern(y, prentice, log = TRUE)))
  } else {
    return(1e100)
  }
}
```

```
likstukel <- function(alpha, beta, y, X) {
  eta <- X %*% beta
  stukel <- plogis(hstukel(eta, alpha))
  stukel <- pmin(pmax(stukel, .Machine$double.eps),
    1 - .Machine$double.eps)
  return(-sum(dbern(y, stukel, log = TRUE)))
}
```

A.3 Funções de Ligação

```
aranda <- function(alpha = 1) {
  linkfun <- function(mu)
    log( ((1 - mu)^(-alpha) - 1)/alpha )
  linkinv <- function(eta)
    ifelse(alpha*exp(eta) > -1, 1 - (alpha*exp(eta) + 1)^(-1/alpha), 1)
  mu.eta <- function(eta)
    ifelse(alpha*exp(eta) > -1,
                        exp(eta)*(alpha*exp(eta) + 1)^(-1/alpha - 1), 0)

  link <- paste("Aranda-Ordaz, alpha =", alpha)
  valideta <- function(eta) TRUE
  structure(list(linkfun = linkfun, linkinv = linkinv,
                 mu.eta = mu.eta, valideta = valideta, name = link),
            class = "link-glm")
}


weibull <- function(gama){
  linkfun <- function(mu)
    return((-log(1 - mu))^(1/gama))
  linkinv <- function(eta) {
    eta <- pmax(eta, 0)
    return(1 - exp(-(eta^gama)))
  }
  mu.eta <- function(eta) {
    eta <- pmax(eta, 0)
    return(gama*eta^(gama - 1)*exp(-eta^gama))
  }
  link <- paste("Weibull, gamma =", gama)
```

```
  valideta <- function(eta)
    return(all(eta > 0))
  structure(list(linkfun = linkfun, linkinv = linkinv,
                 mu.eta = mu.eta, valideta = valideta, name = link),
            class = "link-glm")
}



prentice <- function(m = c(1, 1)) {
  linkfun <- function(mu)  qprentice(mu,m)
  linkinv <- function(eta) pprentice(eta,m)
  mu.eta  <- function(eta) dprentice(eta,m)
  link     <- paste("Prentice, m1 = ", m[1],"  -- m2 = ",m[2],sep="")
  valideta <- function(eta) TRUE
  structure(list(linkfun = linkfun, linkinv = linkinv,
                 mu.eta = mu.eta, valideta = valideta, name = link),
            class = "link-glm")
}



stukel <- function(alpha = c(0, 0)) {
  linkfun  <- function(mu)  invhstukel(qlogis(mu), alpha)
  linkinv  <- function(eta) plogis(hstukel(eta, alpha))
  mu.eta   <- function(eta) dlogis(hstukel(eta, alpha))*dhstukel(eta, alpha)
  link     <- paste("Stukel, a1 = ", alpha[1],"  -- a2 = ",alpha[2], sep = "")
  valideta <- function(eta) TRUE
  structure(list(linkfun = linkfun, linkinv = linkinv,
                 mu.eta = mu.eta, valideta = valideta, name = link),
            class = "link-glm")
}
```

A.4 Função para Estimação Clássica

```
binreg <- function(formula, link = c("Aranda-Ordaz", "Weibull", "Prentice",
          "Stukel"), data, subset, start = NULL, tol = 1e-4, iterlim = 5000,
          na.action){

  call <- match.call()
```

```
if(class(formula) != "formula")
  stop("Invalid formula")
if(missing(data))
  data <- environment(formula)
if(!missing(subset))
  data <- subset(data, subset)
X    <- model.matrix(formula, data)
resp <- model.frame(formula, data)[,1]
resp <- factor(resp)

if(nlevels(resp) > 2)
  warning("Number of levels greater than 2. The first one will be
    regressed against the others")
if(nlevels(resp) < 2)
  stop("Response has fewer than two levels")
resp <- ifelse(resp == levels(resp)[1], 0, 1)

link <- match.arg(link)
i    <- 0
aic  <- 0

if(link == "Aranda-Ordaz"){
  a    <- 1
  fit  <- glm(formula, binomial(aranda(a)), data, start = start)
  bet  <- coef(fit)
  cand <- nlm(likaranda, a, beta = bet, y = resp, X = X)

  while (((abs(cand$estimate - a) > tol) || (abs((cand$estimate
                          - a)/a) > tol)) && (abs(fit$aic - aic) > tol)){
    aic  <- fit$aic
    a    <- cand$estimate
    fit  <- glm(formula, binomial(aranda(a)), data, start = bet)
    bet  <- coef(fit)
    cand <- nlm(likaranda, a, beta = bet, y = resp, X = X, hessian = TRUE)
    i    <- i + 1

    if(i == iterlim){
      warning("Maximum iteration reached. Solution may not be optimal.")
      break
    }
```

```
    }
    a <- cand$estimate
    fit <- glm(formula, binomial(aranda(a)), data, start = bet,
      na.action = na.action)
    fit$link.par <- a
    fit$link.err <- as.numeric(sqrt(1/cand$hessian))
    names(fit$link.par) <- names(fit$link.err) <- "alpha"
    fit$aic <- fit$aic + 2
  }

  if(link == "Weibull"){
    gama <- 3.6
    fit  <- glm(formula, binomial(weibull(gama)), data, start = start)
    bet  <- coef(fit)
    cand <- nlm(likweibull, gama, beta = bet, y = resp, X = X)
    flag <- 0.1

    while(((abs(cand$estimate - gama) > tol) || (abs((cand$estimate
                         - gama)/gama) > tol)) && (abs(fit$aic - aic) > tol)){
      if ((i >= 10) && (abs(fit$aic - aic) < tol) &&
        (all(flag[(length(flag)-10):length(flag)] > 0))){
        warning("Link parameter is too large. Complementary log-log estimate
          returned.")
        fit <- glm(formula, binomial("cloglog"), data, na.action = na.action)
        fit$formula <- formula
        fit$call <- call
        return(fit)
      }

      aic     <- fit$aic
      gama    <- cand$estimate
      fit     <- glm(formula, binomial(weibull(gama)), data, start = bet)
      bet     <- coef(fit)
      cand    <- nlm(likweibull, gama, beta = bet, y = resp, X = X,
        hessian = TRUE)
      i       <- i + 1
      flag[i] <- (cand$estimate - gama)

      if(i == iterlim){
        warning("Maximum iteration reached. Solution may not be optimal.")
```

```
      break
    }
  }

gama <- cand$estimate
fit <- glm(formula, binomial(weibull(gama)), data, start = bet,
  na.action = na.action)
fit$link.par <- gama
fit$link.err <- as.numeric(sqrt(1/cand$hessian))
names(fit$link.par) <- names(fit$link.err) <- "gamma"
fit$aic <- fit$aic + 2
}

if(link == "Prentice"){
  m    <- c(1, 1)
  fit  <- glm(formula, binomial(prentice(m)), data, start = start)
  bet  <- coef(fit)
  cand <- nlm(likprentice, m, beta = bet, y = resp, X = X)
  flag <- c(0.1, 0.1)

  while(((any(abs(cand$estimate - m) > tol)) || (any((abs(cand$estimate
                          - m)/m) > tol))) && (abs(fit$aic - aic) > tol)){
    if ((i >= 10) && (abs(fit$aic - aic) < tol) &&
      ((all(flag[(length(flag)-10):length(flag),1] > 0)) &&
      (all(flag[(length(flag)-10):length(flag),2] > 0)))){
      warning("Link parameters are too large. Probit estimate returned.")
      fit <- glm(formula, binomial("probit"), data, na.action = na.action)
      fit$formula <- formula
      fit$call <- call
      return(fit)
    }

    aic  <- fit$aic
    m    <- cand$estimate
    fit  <- glm(formula, binomial(prentice(m)), data, start = bet)
    bet  <- coef(fit)
    cand <- nlm(likprentice, m, beta = bet, y = resp, X = X, hessian = TRUE)
    i    <- i + 1
    flag <- rbind(flag, (cand$estimate - m))
```

```
    if(i == iterlim){
      warning("Maximum iteration reached. Solution may not be optimal.")
      break
    }
  }

  m <- cand$estimate
  fit <- glm(formula, binomial(prentice(m)), data, start = bet,
    na.action = na.action)
  fit$link.par <- m
  fit$link.err <- as.numeric(sqrt(diag(solve(cand$hessian))))
  names(fit$link.par) <- names(fit$link.err) <- c("m1", "m2")
  fit$aic <- fit$aic + 4
}

if(link == "Stukel"){
  a    <- c(0, 0)
  fit  <- glm(formula, binomial(stukel(a)), data, start = start)
  bet  <- coef(fit)
  cand <- nlm(likstukel, a, beta = bet, y = resp, X = X)

  while(((any(abs(cand$estimate - a) > tol)) || (any(abs((cand$estimate
                          - a)/a) > tol))) && (abs(fit$aic - aic) > tol)){
    aic  <- fit$aic
    a    <- cand$estimate
    fit  <- glm(formula, binomial(stukel(a)), data, start = bet)
    bet  <- coef(fit)
    cand <- nlm(likstukel, a, beta = bet, y = resp, X = X, hessian = TRUE)
    i    <- i + 1

    if(i == iterlim){
      warning("Maximum iteration reached. Solution may not be optimal.")
      break
    }
  }

  a <- cand$estimate
  fit <- glm(formula, binomial(stukel(a)), data, start = bet,
    na.action = na.action)
  fit$link.par <- a
```

```
    fit$link.err <- as.numeric(sqrt(diag(solve(cand$hessian))))
    names(fit$link.par) <- names(fit$link.err) <- c("a1", "a2")
    fit$aic <- fit$aic + 4
  }

  fit$iterations <- i
  fit$formula <- formula
  fit$call <- call
  class(fit) <- c("binreg", class(fit))
  fit
}
```

A.5 Modelos Bayesianos

```
ModelLogit <- function(parm, data, beta.mean = 0, beta.var = 1000){
  Model <- function(parm, data){
    ###parameter
    beta <- parm[1:data$J]
    ###priors
    beta.prior <- dnormv(beta, beta.mean, beta.var, log = TRUE)
    ###logLik
    eta <- data$X %*% beta
    mu <- make.link("logit")$linkinv(eta)
    LL <- sum(dbern(data$y, mu, log = TRUE))
    ###log-posterior
    LP <- LL + sum(beta.prior)
    Modelout <- list(LP = LP, Dev = -2*LL, Monitor = LP, mu = eta,
      parm = parm)
    return(Modelout)
  }
  return(Model)
}

ModelProbit <- function(parm, data, beta.mean = 0, beta.var = 1000){
  Model <- function(parm, data){
    ###parameter
    beta <- parm[1:data$J]
    ###priors
    beta.prior <- dnormv(beta, beta.mean, beta.var, log = TRUE)
```

```
    ###logLik
    eta <- data$X %*% beta
    mu <- make.link("probit")$linkinv(eta)
    LL <- sum(dbern(data$y, mu, log = TRUE))
    ###log-posterior
    LP <- LL + sum(beta.prior)
    Modelout <- list(LP = LP, Dev = -2*LL, Monitor = LP, mu = eta,
      parm = parm)
    return(Modelout)
  }
  return(Model)
}

ModelCloglog <- function(parm, data, beta.mean = 0, beta.var = 1000){
  Model <- function(parm, data){
    ###parameter
    beta <- parm[1:data$J]
    ###priors
    beta.prior <- dnormv(beta, beta.mean, beta.var, log = TRUE)
    ###logLik
    eta <- data$X %*% beta
    mu <- make.link("cloglog")$linkinv(eta)
    LL <- sum(dbern(data$y, mu, log = TRUE))
    ###log-posterior
    LP <- LL + sum(beta.prior)
    Modelout <- list(LP = LP, Dev = -2*LL, Monitor = LP, mu = eta,
      parm = parm)
    return(Modelout)
  }
  return(Model)
}

ModelAranda <- function(parm, data, beta.mean = 0, beta.var = 1000,
                        alpha.scale = 1e-10){
   Model <- function(parm, data){
    ###parameter
    beta <- parm[1:data$J]
    alpha <- parm[data$J + 1]
    ###priors
    beta.prior <- dnormv(beta, beta.mean, beta.var, log = TRUE)
```

```
    alpha.prior <- dhalfnorm(alpha, alpha.scale, log = TRUE)
    ###logLik
    LL <- -likaranda(alpha, beta, data$y, data$X)
    ###log-posterior
    LP <- LL + sum(beta.prior) + alpha.prior
    Modelout <- list(LP = LP, Dev = -2*LL, Monitor = LP, mu = data$X %*% beta,
      parm = parm)
    return(Modelout)
   }
   return(Model)
}

ModelWeibull <- function(parm, data, beta.mean = 0, beta.var = 1000,
                         gamma.scale = 1e-10){
  Model <- function(parm, data){
    ###parameter
    beta <- parm[1:data$J]
    gamma <- parm[data$J + 1]
    ###priors
    beta.prior <- dnormv(beta, beta.mean, beta.var, log = TRUE)
    gamma.prior <- dhalfnorm(gamma, gamma.scale, log = TRUE)
    ###logLik
    LL <- -likweibull(gamma, beta, data$y, data$X)
    ###log-posterior
    LP <- LL + sum(beta.prior) + gamma.prior
    Modelout <- list(LP = LP, Dev = -2*LL, Monitor = LP, mu = data$X %*% beta,
      parm = parm)
    return(Modelout)
  }
  return(Model)
}

ModelPrentice <- function(parm, data, beta.mean = 0, beta.var = 1000,
                          m.scale = 1e-10){
  Model <- function(parm, data){
    ###parameter
    beta <- parm[1:data$J]
    m <- parm[c(data$J + 1, data$J + 2)]
    ###priors
    beta.prior <- dnormv(beta, beta.mean, beta.var, log = TRUE)
```

```
    m.prior <- dhalfnorm(m, m.scale, log = TRUE)
    ###logLik
    LL <- -likprentice(m, beta, data$y, data$X)
    ###log-posterior
    LP <- LL + sum(beta.prior) + sum(m.prior)
    Modelout <- list(LP = LP, Dev = -2*LL, Monitor = LP, mu = data$X %*% beta,
      parm = parm)
    return(Modelout)
  }
  return(Model)
}

ModelStukel <- function(parm, data, beta.mean = 0, beta.var = 1000,
                        alpha.mean = 0, alpha.var = 1000){
  Model <- function(parm, data){
    ###parameter
    beta <- parm[1:data$J]
    alpha <- parm[c(data$J + 1, data$J + 2)]
    ###priors
    beta.prior <- dnormv(beta, beta.mean, beta.var, log = TRUE)
    alpha.prior <- dnormv(alpha, alpha.mean, alpha.var, log = TRUE)
    ###logLik
    LL <- -likstukel(alpha, beta, data$y, data$X)
    ###log-posterior
    LP <- LL + sum(beta.prior) + sum(alpha.prior)
    Modelout <- list(LP = LP, Dev = -2*LL, Monitor = LP, mu = data$X %*% beta,
      parm = parm)
    return(Modelout)
  }
  return(Model)
}
```

A.6 Funções Extras

```
logLik.binreg <- function(object, ...){
  if (length(list(...)))
    warning("extra arguments discarded")

  link <- strtrim(object$family$link, 1)
```

```
  p <- object$rank + switch(link, A = 1, W = 1, P = 2, S = 2)
x   val <- p - object$aic/2
  attr(val, "nobs") <- sum(!is.na(object$residuals))
  attr(val, "df") <- p
  class(val) <- "logLik"
  val
}

summary.binreg <- function(object, ...){
  ans <- NextMethod("summary")
  link <- strtrim(object$family$link, 1)

  if(link == "A"){
    zval <- (object$link.par - 1)/object$link.err
    pval <- pchisq(zval^2, 1, lower.tail = FALSE)
    ans$coefficients <- rbind(ans$coefficients, alpha = c(object$link.par,
      object$link.err, zval, pval))
  }

  if(link == "W"){
    zval <- (object$link.par - 3.50215)/object$link.err
    pval <- pchisq(zval^2, 1, lower.tail = FALSE)
    ans$coefficients <- rbind(ans$coefficients, gamma = c(object$link.par,
      object$link.err, zval, pval))
  }

  if(link == "P"){
    zval <- (object$link.par - 1)/object$link.err
    pval <- pchisq(zval^2, 1, lower.tail = FALSE)
    ans$coefficients <- rbind(ans$coefficients, m1 = c(object$link.par[1],
      object$link.err[1], zval[1], pval[1]), m2 = c(object$link.par[2],
      object$link.err[2], zval[2], pval[2]))
  }

  if(link == "S"){
    zval <- object$link.par/object$link.err
    pval <- pchisq(zval^2, 1, lower.tail = FALSE)
    ans$coefficients <- rbind(ans$coefficients, a1 = c(object$link.par[1],
      object$link.err[1], zval[1], pval[1]), a2 = c(object$link.par[2],
      object$link.err[2], zval[2], pval[2]))
```

```
  }

  return(ans)
}

print.binreg <- function (x, digits = max(3, getOption("digits") - 3), ...){
  cat("\nCall:  ", paste(deparse(x$call), sep = "\n", collapse = "\n"),
      "\n\n", sep = "")
  if (length(coef(x))) {
    cat("Coefficients")
    if (is.character(co <- x$contrasts))
      cat("  [contrasts: ", apply(cbind(names(co), co),
                                  1L, paste, collapse = "="), "]")
    cat(":\n")
    print.default(format(x$coefficients, digits = digits),
                  print.gap = 2, quote = FALSE)
  }
  else cat("No coefficients\n\n")

  cat("\nLink Parameters:\n")
  print.default(format(x$link.par, digits = digits), print.gap = 2,
    quote = FALSE)

  cat("\nDegrees of Freedom:", x$df.null, "Total (i.e. Null); ",
    x$df.residual, "Residual\n")
  if (nzchar(mess <- naprint(x$na.action)))
    cat("  (", mess, ")\n", sep = "")
  cat("Null Deviance:\t   ", format(signif(x$null.deviance, digits)),
      "\nResidual Deviance:", format(signif(x$deviance, digits)),
      "\tAIC:", format(signif(x$aic, digits)), "\n")
  invisible(x)
}

compareIC <- function(...){
  names <- as.character(match.call())[-1]
  objects <- list(...)

  df <- sapply(objects, function(x) attr(logLik(x), "df"))
  llik <- sapply(objects, logLik)
  aic <- sapply(objects, AIC)
```

```
  bic <- sapply(objects, BIC)

  data.frame(df = df, logLik = llik, AIC = aic, BIC = bic, row.names = names)
}

LRTlogit <- function(object){
  if(!inherits(object, "binreg"))
    stop("The object is not from class binreg")

  dname <- deparse(substitute(object))
  logit <- glm(object$formula, binomial, object$data)
  lr <- 2*as.numeric(logLik(object) - logLik(logit))
  df <- attr(logLik(object), "df") - attr(logLik(logit), "df")
  pval <- pchisq(lr, df, lower.tail = FALSE)
  method <- "Likelihood ratio test for Binary Regression with parametric link"

  RVAL <- list(statistic = c(LR = lr), parameter = c(df = df), p.value = pval,
               method = method, data.name = dname)
  class(RVAL) <- "htest"
  return(RVAL)
}

dataLD <- function(formula, link = c("logit", "probit", "cloglog",
    "Aranda-Ordaz", "Weibull", "Prentice", "Stukel"), data, subset){
  formula <- formula(formula)
  if(missing(data))
    data <- environment(formula)
  if(!missing(subset))
    data <- subset(data, subset)
  X    <- model.matrix(formula, data)
  resp <- model.frame(formula, data)[,1]
  resp <- factor(resp)

  if(nlevels(resp) > 2)
    warning("Number of levels greater than 2. The first one will be regressed
      against the others.")
  if(nlevels(resp) < 2)
    stop("Response has fewer than two levels.")
  resp <- ifelse(resp == levels(resp)[1], 0, 1)
```

```
link <- match.arg(link)

if(link == "logit"){
  parm.names <- as.parm.names(list(beta = rep(0, ncol(X))))
  model <- "ModelLogit()"
}

if(link == "probit"){
  parm.names <- as.parm.names(list(beta = rep(0, ncol(X))))
  model <- "ModelProbit()"
}

if(link == "cloglog"){
  parm.names <- as.parm.names(list(beta = rep(0, ncol(X))))
  model <- "ModelCloglog()"
}

if(link == "Aranda-Ordaz"){
  parm.names <- as.parm.names(list(beta = rep(0, ncol(X)), alpha = 0))
  model <- "ModelAranda()"
}

if(link == "Weibull"){
  parm.names <- as.parm.names(list(beta = rep(0, ncol(X)), gamma = 0))
  model <- "ModelWeibull()"
}

if(link == "Prentice"){
  parm.names <- as.parm.names(list(beta = rep(0, ncol(X)), m = c(0, 0)))
  model <- "ModelPrentice()"
}

if(link == "Stukel"){
  parm.names <- as.parm.names(list(beta = rep(0, ncol(X)), a = c(0, 0)))
  model <- "ModelStukel()"
}

dataset <- list(J = ncol(X), X = X, mon.names = "LP",
  parm.names = parm.names, y = resp)
```

```
  cat("Demonic dataset is ready!\n\n")
  print(data.frame(variable = colnames(X), parameter = parm.names[1:ncol(X)]),
    row.names = F)
  cat("\nLaplace's Demon suggests running the following code:\n\n")
  cat("fit = LaplacesDemon(", model, ", data, GIV(", model, ", data))\n\n",
    sep = "")
  cat("Remember to substitute 'data' with the object returned by this
    function!")

  invisible(dataset)
}
```

Referências Bibliográficas

Abramowitz e Stegun (1965) M. Abramowitz e I. A. Stegun, editores. *Handbook of Mathematical Functions*. New York: Dover. Citado na pág. 15

Agresti (2002) A. Agresti. *Categorical Data Analysis*. Wiley-Interscience, 2ª ed. Citado na pág. 1, 25

Akaike (1974) H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723. Citado na pág. 17

Aranda-Ordaz (1981) F. J. Aranda-Ordaz. On two families of transformations to additivity for binary response data. *Biometrika*, 68(2):357–363. Citado na pág. 1, 16

Balasem e Ali (1991) A. N. Balasem e A. S. K. Ali. Establishment of dose-response relationships between doses of Cs-137 γ-rays and frequencies of micronuclei in human peripheral blood lymphocytes. *Mutation Research*, 259(2):133–138. Citado na pág. 35, 43

Bliss (1934) C. I. Bliss. The method of probits. *Science*, 79:38–39. Citado na pág. 3

Bliss (1935) C. I. Bliss. The calculation of the dosage-mortality curve. *Annals of Applied Biology*, 22:134–167. Citado na pág. 35

Caron (2010) R. Caron. Regressão de dados binários: Distribuição weibull. Dissertação de Mestrado, Universidade Federal de São Carlos. Citado na pág. 2, 8

Caron *et al.* (2009) R. Caron, A. Polpo, P. M. Goggans e C. Y. Chan. Binary data regression: Weibull distribution. Em *Aip Conference Proceedings*, volume 1193. Citado na pág. 2

Cramer (2002) J. S. Cramer. The origins of logistic regression. 2002. Citado na pág. 3

DeGroot (2004) M. H. DeGroot. *Optimal Statistical Decisions*. Wiley-Interscience. Citado na pág. 41

Dennis e Schnabel (1996) J. E. Dennis e R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*, volume 16. Society for Industrial Mathematics. Citado na pág. 13

Duane *et al.* **(1987)** S. Duane, A. D. Kennedy, B. J. Pendleton e D. Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222. Citado na pág. 29

Fisher (1956) R. A. Fisher. *Statistical Methods and Scientific Inference*. Hafner Publishing Co. Citado na pág. 16

Geman e Geman (1984) S. Geman e D. Geman. Stochastic relaxation, Gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):721–741. Citado na pág. 29

Geweke (1992) J. Geweke. Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. Em J. M. Bernardo, J. O. Berger, A. P. Dawid e A. F. M. Smith, editores, *Bayesian Statistics*. Clarendon Press. Citado na pág. 33

Haario *et al.* **(2001)** H. Haario, E. Saksman e J. Tamminen. An adaptive Metropolis algorithm. *Bernoulli*, páginas 223–242. Citado na pág. 40

Hall (2012) B. Hall. *LaplacesDemon: Complete Environment for Bayesian Inference*, 2012. URL `http://cran.r-project.org/web/packages/LaplacesDemon/index.html`. R package version 12.09.03. Citado na pág. 29

Hastings (1970) W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109. Citado na pág. 29

Kutner *et al.* **(2005)** M. H. Kutner, C. J. Nachtsheim, J. Neter e W. Li. *Applied Linear Statistical Models*. McGraw Hill, 5ª ed. Citado na pág. 1

Nelder e Wedderburn (1972) J. A. Nelder e R. W. M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)*, 135(3):370–384. Citado na pág. 1

Osborn e Madey (1968) D. Osborn e R. Madey. The incomplete beta function and its ratio to the complete beta function. *Mathematics of Computation*, 22:159–162. Citado na pág. 15

Prentice (1975) R. L. Prentice. Discrimination among some parametric models. *Biometrika*, 62(3):607–614. Citado na pág. 1, 8, 16

Prentice (1976) R. L. Prentice. A generalization of the probit and logit methods for dose response curves. *Biometrics*, páginas 761–768. Citado na pág. 2, 14, 16

R Core Team (2012) R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. URL `http://www.R-project.org/`. Citado na pág. 2, 13

Roberts *et al.* **(1997)** G. O. Roberts, A. Gelman e W. R. Gilks. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7(1):110–120. Citado na pág. 41

Schwarz (1978) G. Schwarz. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464. Citado na pág. 24

Spiegelhalter *et al.* **(2002)** D. J. Spiegelhalter, N. G. Best, Bradley P. C. e A. Van Der Linde. Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(4):583–639. Citado na pág. 40

Stukel (1988) T. A. Stukel. Generalized logistic models. *Journal of the American Statistical Association*, páginas 426–431. Citado na pág. 1, 9, 16

Wilks (1938) S. S. Wilks. The large-sample distribution of the likelihood ratio for testing composite hypotheses. *The Annals of Mathematical Statistics*, 9:60–62. Citado na pág. 27