Métodos computacionais para inferência estatística

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

2 de março de 2012

Capítulo 1

Introdução

A abordagem estatística para análise e resumo de informações contidas em um conjunto de dados, consiste na suposição de que existe um mecanismo estocástico gerador do processo em análise. Este mecanismo é descrito através de um modelo probabilístico, representado por uma distribuição de probabilidade. Em situações reais a verdadeira distribuição de probabilidade geradora do processo é desconhecida, sendo assim, distribuições de probabilidade adequadas devem ser escolhidas de acordo com o tipo de fenômeno em análise. Por exemplo, se o fenômeno em estudo consiste em medir uma característica numérica de um grupo de indivíduos em uma escala contínua, distribuições com este **suporte** devem ser escolhidas. O **suporte** de uma distribuição de probabilidade informa qual o domínio da função, ou seja, quais são os valores que a variável aleatória pode assumir. Considere o caso da distribuição Gaussiana, o **suporte** é a reta real, no caso da distribuição Gama o suporte é apenas os reais positivos. Um cuidado adicional deve ser dado quando a variável de interesse é discreta, por exemplo contagens, onde é comum atribuir uma distribuição de Poisson que tem **suporte** nos naturais positivos.

Em quase todos os problemas de modelagem estatística existem duas ou mais distribuições de probabilidade candidatas à representar o fenômeno. Porém, na maioria das situações assume-se que a distribuição de probabilidade geradora do processo é conhecida, com excessão dos valores de um ou mais **parâmetros** que a indexam. Por exemplo, considere que o tempo de vida de um tipo de reator nuclear tem distribuição exponencial com **parâmetro** λ, mas o valor exato de λ é desconhecido. Se o tempo de vida de vários reatores nuclear de mesmo tipo são observados, com estes dados e qualquer outra fonte relevante de informação que esteja disponível, é possível fazer **inferência** sobre o valor desconhecido do parâmetro λ. O processo de **inferência** consiste em encontrar um valor mais plausível de ser λ, bem como, informar um intervalo para o qual acredita-se conter o verdadeiro valor de λ, além de decidir ou opinar se λ é igual, maior ou menor que algum valor previamente especificado.

Em implementações computacionais para inferência estatística, deve-se sempre estar atento ao **espaço paramétrico** (Θ) de um modelo probabilístico. No caso, do tempo de vida de reatores nucleares, assumindo que a distribuição exponencial é adequada e esta sendo indexada pelo parâmetro λ, de acordo com a construção do modelo exponencial, tem-se que o **espaço paramétrico** de λ é os reais positivos. Em outras situações, por exemplo de um modelo Normal, com

média μ e variância σ^2, tem-se que o **espaço paramétrico** para μ é os reais, enquanto que para σ^2 é os reais positivos. Estas restrições precisam ser levadas em consideração no processo de inferência e são de fundamental importância para o sucesso de muitos algoritmos de maximização numérica.

Partindo destes conceitos, um fenômeno aleatório ou estocástico é descrito minimamente por uma distribuição de probabilidade, que por sua vez é descrita por seus parâmetros e respectivos campos de variação **(espaço paramétrico)**. Além, do campo de variação da própria variável aleatória que deve ser compatível com o suporte da distribuição atribuída ao fenômeno. Por exemplo, não é correto atribuir uma distribuição de Poisson para a altura (medida contínua) de trabalhadores, uma vez que o campo de variação da variável de interesse (resposta) não é compatível com o suporte da distribuição de probabilidade.

Considere o caso onde deseja-se fazer uma pesquisa a respeito da intenção de voto em um determinado candidato. Suponha que n eleitores obtidos aleatóriamente são questionados sobre a sua intenção em votar (1) ou não votar (0) em determinado candidato. Neste caso, tem-se como possíveis resultados do questionamento a resposta (1) ou (0). Deseja-se saber a probabilidade de um eleitor ao acaso manisfetar a intenção de votar neste candidato. Dado a estrutura do experimento, pode-se supor que o modelo de Bernoulli seja adequado para esta situação. Este modelo tem como suporte o 0 e 1, compatível com o experimento, além disso, tem como seu parâmetro indexador p que representa a probabilidade de sucesso, ou seja, votar no candidato, para completar a especificação este parâmetro tem como seu **espaço paramétrico** o intervalo unitário. Com este conjunto de suposições e conhecimentos a respeito do modelo probabilístico, tem-se total condições de fazer **inferência** sobre o parâmetro p.

Neste livro será dada ênfase ao método de máxima verossimilhança, que fornece **estimadores** com propriedades convenientes para os parâmetros desconhecidos de um modelo probabilístico. Este método é baseado na **função de verossimilhança**, e fornece uma abordagem integrada para a obtenção de estimativas pontuais e intervalares, além da construção de testes de hipóteses. Toda a metodologia será descrita através de exemplos abordando diversos aspectos teóricos, com ênfase em como implementar a estimação de parâmetros desconhecidos desde os modelos mais simples até modelos altamente estruturados.

1.1 Estimação pontual

Seja $Y_1, Y_2, \ldots, Y_n$ variáveis aleatórias com função probabilidade ou densidade probabilidade, denotada por $f(\underline{Y}, \underline{\theta})$ para os casos discreto e contínuo respectivamente, onde $\underline{\theta}$ é um vetor de parâmetros desconhecidos (um único elemento de $\underline{\theta}$ será denotado por θ), aos quais se deseja estimar através de amostras $y_1, y_2, \ldots, y_n$ realizações das variáveis aleatórias $Y_1, Y_2, \ldots, Y_n$. Para denotar esta situação $Y_i \sim f(\underline{\theta})$ com $i = 1, \ldots, n$.

Definição 1 (Estatística)**.** *Uma **estatística** é uma variável aleatória $T = t(\underline{Y})$, onde a função $t(.)$ não depende de θ.*

Definição 2 (Estimador)**.** *Uma estatística T é um estimador para θ se o valor realizado $t = t(\underline{y})$ é usado como uma estimativa para o valor de θ.*

Definição 3 (Distribuição amostral)**.** *A distribuição de probabilidade de T é chamada de* ***distribuição amostral*** *do estimador $t(\underline{Y})$.*

Definição 4 (Viés)**.** *O* ***viés*** *de um estimador T é a quantidade*

$$B(T) = E(T - \theta).$$

O estimador T é dito não viciado para θ se $B(T) = 0$, tal que $E(T) = \theta$. O estimador T é assintóticamente não viciado para θ se $E(T) \to \theta$ quando $n \to \infty$.

Definição 5 (Eficiência relativa)**.** *A* ***eficiência relativa*** *entre dois estimadores T_1 e T_2 é a razão $er = \frac{V(T_1)}{V(T_2)}$.*

Definição 6 (Erro quadrático médio)**.** *O* ***erro quadrático médio*** *de um estimador T é a quantidade*

$$EQM(T) = E((T - \theta)^2) = V(T) + B(T)^2$$

Definição 7 (Consistência)**.** *Um estimador T é* ***médio quadrático consistente*** *para θ se o $EQM(T) \to 0$ quando $n \to \infty$. O estimador T é* ***consistente em probabilidade*** *se $\forall \epsilon > 0$, $P(|T - \theta| > \epsilon) \to 0$, quando $n \to \infty$.*

Estas definições introduzem conceitos e propriedades básicas para uma estatística ser um estimador adequado para um determinado parâmetro. Fracamente falando, o desejo é obter um estimador que seja assintóticamente não-viciado, ou seja, conforme o tamanho da amostra aumenta ele se aproxima cada vez mais do verdadeiro valor do parâmetro. Além disso, é interessante que ele seja eficiente, ou seja, apresente a menor variância possível entre todos os estimadores de θ. Esta definição de eficiência, introduz o conceito de variância minima. Sendo assim, para saber se um estimador é eficiente é necessário conhecer um limite inferior para a variância de um estimador, uma vez que tal quantidade existe e seja passível de calcular, ao propor um estimador para θ, basta calcular a sua variância e comparar com a menor possível, se ele atingir este limite será eficiente. Além disso, tomando sua esperança pode-se concluir sobre o seu viés dependendo da situação em termos assintóticos. O Teorema 1, ajuda a responder sobre a eficiência de um estimador qualquer. Mas antes precisamos de mais algumas definições.

Definição 8 (Verossimilhança)**.** *Suponha que os dados $\underline{y}$ são uma realização de um vetor aleatório $\underline{Y}$ com função de probabilidade ou densidade probabilidade $f(\underline{Y}, \underline{\theta})$. A* ***verossimilhança*** *para $\underline{\theta}$ dado os valores observados $\underline{y}$ é a função $L(\underline{\theta}, \underline{y})$.*

A função de verossimilhança é a distribuição conjunta de todas as variáveis aleatórias envolvidas no modelo, porém olhando-a como função dos parâmetros, uma vez que os dados já foram observados e portanto são quantidades fixas. Pense na verossimilhança como uma medida de compatibilidade da amostra com o particular conjunto de parâmetros que está sendo avaliado, para medir a sua compatibilidade ou plausibilidade ou similaridade com a amostra observada. No caso onde os elementos de $\underline{y}$ são independentes a verossimilhança é simplesmente um produto das distribuições de cada amostra individual, ou seja, $L(\underline{\theta}, \underline{y}) = \prod_{i=1}^{n} f(y_i, \underline{\theta})$. Neste caso, o procedimento de inferência pode ser bastante facilitado, porém cabe ressaltar que isto não é uma exigência, e situações

onde as amostras não são independentes podem ser tratadas no mesmo *framework* geral, tendo apenas o cuidado de escrever a verossimilhança de uma forma adequada.

O particular valor assumido pela função de verossimilhança não é importante, o que interessa para **inferência** são os valores relativos de $L(\underline{\theta}, \underline{y})$ para diferentes conjuntos de $\underline{\theta}$. Como dito, a verossimilhança é uma medida de compatibilidade da amostra observada com um particular vetor de parâmetros, desta forma é natural definir como estimador para o vetor de parâmetros $\underline{\theta}$, aquele particular vetor digamos $\hat{\underline{\theta}}$ que tenha mais compatibilidade com a amostra, ou em outras palavras o vetor que maximiza a função de verossimilhança ou compatibilidade.

Definição 9. *Seja $L(\underline{\theta}, \underline{y})$ a função de verossimilhança. O valor $\hat{\underline{\theta}} = \hat{\underline{\theta}}(\underline{y})$ é a estimativa de máxima verossimilhança para $\underline{\theta}$ se $L(\hat{\underline{\theta}}) \geq L(\underline{\theta})$, $\forall \underline{\theta}$.*

Definição 10. *Se $\hat{\underline{\theta}}(\underline{y})$ é a estimativa de máxima verossimilhança, então $\hat{\underline{\theta}}(\underline{Y})$ é o estimador de máxima verossimilhança.*

Nesta etapa é preciso ter cuidado com a notação. Veja que $\hat{\underline{\theta}}(\underline{y})$ é um vetor de escalares, enquanto que $\hat{\underline{\theta}}(\underline{Y})$ é um vetor de variáveis aleatórias. Daqui em diante usaremos apenas $\hat{\underline{\theta}}$, para ambos sendo que o contexto indicará o real sentido de $\hat{\underline{\theta}}$. A função de verossimilhança contêm toda a informação proveniente dos dados sobre o vetor de parâmetros $\underline{\theta}$. Apesar disso, a $L(\underline{\theta})$ é computacionalmente incoveniente, uma vez que esta função apresentará valores muito próximos de zero. Por razões meramente computacionais é mais comum usar a função de log-verossimilhança, definida por:

Definição 11 (Log-verossimilhança)**.** *Se $L(\underline{\theta})$ é a função de verossimilhança, então $l(\underline{\theta}) = \log L(\underline{\theta})$ é a função de log-verossimilhança.*

Segue do fato da função log ser monotona crescente que maximizar $L(\underline{\theta})$ e $l(\underline{\theta})$ levam ao mesmo ponto de máximo. Neste ponto estamos habilitados a enunciar um dos teoremas mais fortes da teoria de verossimilhança.

Teorema 1 (Limite inferior de Cramer Rao)**.** *Se T é um estimador não-viciado para θ e $l(\theta, \underline{Y})$ é duas vezes diferenciável com respeito a θ, então*

$$V(T) \geq \frac{1}{E(-l''(\theta, \underline{Y}))}$$

Este teorema informa o limite inferior para a variância de um estimador T qualquer. Na sequência veremos que o estimador de máxima verossimilhança apresenta propriedades ótimas e uma delas é a eficiência, ou seja, assintóticamente o EMV atinge o limite inferior de Cramer-Rao. Antes de discutirmos as propriedades dos estimadores de máxima verossimilhança, vamos apresentar uma forma de introduzir a incerteza associada a estimativa de um parâmetro qualquer. Lembre-se que o estimador é um variável aleatória, a estimativa é uma realização desta variável aleatória. Sendo assim, quando reportamos apenas a estimativa pontual, estamos ignorando a incerteza associada a esta estimativa. Uma forma, tradicional de se medir e informar a incerteza associada é com a construção de intervalos de confiança, este é o assunto da próxima Seção.

1.2 Intervalos de confiança baseado em verossimilhança

Definição 12 (Intervalo de confiança)**.** *Um intervalo de verossimilhança para θ é um intervalo da forma $\theta : L(\theta) \geq rL(\hat{\theta})$ ou equivalentemente, $\theta : D(\theta) \geq c$, onde $D(\theta) = 2(l(\hat{\theta}) - l(\theta))$ e $c = -2log(r)$.*

Esta definição é bastante geral para o caso uni-paramétrico, para o caso multi-parâmetros precisamos trocar o intervalo de confiança por uma região de confiança que será abordado mais a frente. Nesta definição o valor de r precisa ser especificado entre 0 e 1, para intervalos não vazios, logo $c > 0$. Quanto maior o valor de c mais largo será o intervalo, alguma vezes o intervalo pode ser a união de sub-intervalos disjuntos. Apesar do valor de c ser a chave para a construção dos intervalos ainda não temos condições de especificar um valor para ele.

Usando esta definição pode-se pensar em duas formas básicas de construção de intervalos de confiança. A primeira é olhar a quantidade $\frac{L(\theta)}{L(\hat{\theta})} \geq r$ que é a verossimilhança relativa, ou seja, compara cada valor de θ com o máximo. Nestas condições a verossimilhança relativa toma sempre valores entre 0 e 1, então pode-se por exemplo, definir $r = 0.8$, ou seja, estamos deixando que faça parte do intervalo de confiança valores que tenham até 80% de compatibilidade com a amostra, da mesma forma poderiamos definir $r = 0.20$ ou 0.5, dependendo de nosso critério.

Uma forma equivalente é olhar para a função *deviance* dada por $D(\theta) = 2(l(\hat{\theta}) - l(\theta)) \geq -2log(r)$ que é uma outra forma de olhar a verossimilhança relativa, em termos de diferença em log-verossimilhança. Novamente surge o problema de definir o valor de $c = -2log(r)$, e com isto encontrar as raízes da função *deviance* que fornecem os limites do intervalo de confiança para um c qualquer especificado. Encontrar as raízes da função *deviance* comumente envolve métodos numéricos, uma vez que na maioria das situações práticas não é possível obter uma solução explícita para o intervalo. Note que esta também é uma restrição para a construção de intervalos baseados em perda relativa de verossimilhança.

Dado esta restrição é comum fazer uma expansão em séries de Taylor para a $l(\theta)$ em torno de $\hat{\theta}$ de forma a facilitar a obtenção do intervalo de confiança.

$$D(\theta) = 2[l(\hat{\theta}) - l(\theta)] = 2[l(\hat{\theta}) - [l(\hat{\theta}) + (\theta - \hat{\theta})l'(\hat{\theta}) + \frac{1}{2}(\theta - \hat{\theta})^2 l''(\hat{\theta})]].$$

Lembrando que $l'(\hat{\theta}) = 0$ e eliminando os termos chegamos a

$$D(\theta) = [-(\theta - \hat{\theta})^2 l''(\hat{\theta})] \geq c.$$

Que leva a intervalos de confiança da forma,

$$\tilde{\theta} = \hat{\theta} \pm \sqrt{\frac{c}{l''(\hat{\theta})}}.$$

Isto corresponde a fazer uma aproximação quadrátiva da função *deviance*, que torna o intervalo fácil de ser obtido. Extendendo para o caso multiparâmetros, tem-se que uma região de confiança para $\underline{\theta}$ é dada pelo conjunto $\underline{\theta} \in \Theta : D(\underline{\theta} \geq c^*)$,

as duas formas de olhar o intervalo de confiança uniparamétrico podem ser extendidas para o caso multiparamétrico, sem problemas. Novamente o problema que surge é a obtença do valor de c^*. Pela abordagem frequentista é desejável que o intervalo tenha uma interpretação em termos de probabilidades ou frequência e isto é atingido através das propriedades assintóticas dos estimadores de máxima verossimilhança, que serão apresentadas na próxima Seção.

1.3 Propriedades dos estimadores de máxima verossimilhança

Apesar de olharmos a função de verossimilhança como uma quantidade fixa avaliada em $\underline{y}$, devemos lembrar que ela é baseada em apenas uma realização do vetor aleatório $\underline{Y}$, sendo assim, estudar o comportamento probabilístico dos estimadores de máxima verossimilhança é de fundamental importância para a construção de intervalos de confiança e testes de hipóteses. Para isto, vamos precisar de mais algumas definições.

Definição 13 (Função escore)**.** *Sendo $l(\underline{\theta})$ a função de log-verossimilhança, o vetor escore é definido por*

$$U(\underline{\theta}) = \left(\frac{\partial l(\underline{\theta})}{\partial \theta_1}, \ldots, \frac{\partial l(\underline{\theta})}{\partial \theta_d} \right)^\top,$$

é o vetor gradiente da função de log-verossimilhança.

Definimos as matrizes de informação Observado e Esperado (matriz de informação de Fisher).

Definição 14 (Matriz de informação Observada)**.** *Sendo $l(\underline{\theta})$ a função de log-verossimilha, a matriz de informação Observada é definida por*

$$I_O(\underline{\theta}) = \begin{bmatrix} -\frac{\partial^2 l(\underline{\theta})}{\partial \theta_1^2} & \cdots & \cdots & -\frac{\partial^2 l(\underline{\theta})}{\partial \theta_1 \partial \theta_d} \\ \vdots & \ddots & -\frac{\partial^2 l(\underline{\theta})}{\partial \theta_i \partial \theta_j} & \vdots \\ \vdots & -\frac{\partial^2 l(\underline{\theta})}{\partial \theta_j \partial \theta_i} & \ddots & \vdots \\ -\frac{\partial^2 l(\underline{\theta})}{\partial \theta_d \partial \theta_1} & \cdots & \cdots & -\frac{\partial^2 l(\underline{\theta})}{\partial \theta_d^2} \end{bmatrix}.$$

Definição 15 (Matriz de informação Esperada)**.** *Sendo $l(\underline{\theta})$ a função de log-verossimilha, a matriz de informação Esperada é definida por*

$$I_E(\underline{\theta}) = \begin{bmatrix} E\left[-\frac{\partial^2 l(\underline{\theta})}{\partial \theta_1^2}\right] & \cdots & \cdots & E\left[-\frac{\partial^2 l(\underline{\theta})}{\partial \theta_1 \partial \theta_d}\right] \\ \vdots & \ddots & E\left[-\frac{\partial^2 l(\underline{\theta})}{\partial \theta_i \partial \theta_j}\right] & \vdots \\ \vdots & E\left[-\frac{\partial^2 l(\underline{\theta})}{\partial \theta_j \partial \theta_i}\right] & \ddots & \vdots \\ E\left[-\frac{\partial^2 l(\underline{\theta})}{\partial \theta_d \partial \theta_1}\right] & \cdots & \cdots & E\left[-\frac{\partial^2 l(\underline{\theta})}{\partial \theta_d^2}\right] \end{bmatrix}.$$

Dois importantes resultados da função *escore* e da matriz de informação observada é que $E[U(\underline{\theta})] = 0$ e $V[U(\underline{\theta})] = E[I_O(\underline{\theta})] = I_E[\underline{\theta}]$.

Note que a variância do vetor $U(\underline{\theta})$ é a matriz com entradas

$$\begin{bmatrix} Cov(U_1,U_1) & \dots & \dots & Cov(U_1,U_d) \\ \vdots & \ddots & Cov(U_i,U_j) & \vdots \\ \vdots & Cov(U_j,U_i) & \ddots & \vdots \\ Cov(U_d,U_1) & \dots & \dots & Cov(U_d,U_d) \end{bmatrix}.$$

onde $Cov(U_i,U_i) = V(U_i)$. Uma propriedade importante de $I_O(\hat{\underline{\theta}})$ e $I_E(\hat{\underline{\theta}})$ é que elas são matrizes definida positiva, as quais mensuram a curvatura observada/esperada na superfície de log-verossimilhança. Com estas definições, pode-se escrever a função *deviance* aproximada em termos multiparâmetros da seguinte forma:

$$D(\underline{\theta}) \approx (\underline{\theta} - \hat{\underline{\theta}})^\top I_O(\hat{\underline{\theta}})(\underline{\theta} - \hat{\underline{\theta}})$$

assim $D(\underline{\theta})$ será positiva desde que $I_O(\hat{\underline{\theta}})$ seja uma matriz positiva definida. Uma vez definida todas as quantidades envolvidas na situação, estamos aptos a enunciar o seguinte Teorema:

Teorema 2 (Distribuição assintótico do EMV)**.** *Para um problema de estimação regular, no limite com $n \to \infty$, se $\underline{\theta}$ é o verdadeiro vetor de parâmetros, então*

$$\hat{\underline{\theta}} \sim NM_d(\underline{\theta}, I_E(\theta)^{-1}),$$

ou seja, a distribuição assintótica de $\hat{\underline{\theta}}$ é uma normal multivariada com matriz de variância/covariância dada pela inversa da matriz de informação esperada.

Corolário - Qualquer termo assintoticamente equivalente a $I_E(\underline{\theta})$ pode ser usado no Teorema 1.6. Assim,

$$\hat{\underline{\theta}} \sim NM_d(\underline{\theta}, I_E(\hat{\underline{\theta}})^{-1})$$

$$\hat{\underline{\theta}} \sim NM_d(\underline{\theta}, I_O(\underline{\theta})^{-1})$$

$$\hat{\underline{\theta}} \sim NM_d(\underline{\theta}, I_O(\hat{\underline{\theta}})^{-1}).$$

Teorema 3 (Distribuição assintótica da Deviance)**.** *Para um problema regular de estimação, no limite com $n \to \infty$, se $\underline{\theta}$ é o verdadeiro valor do parâmetro, então*

$$D(\underline{\theta}) = 2[l(\hat{\underline{\theta}}) - l(\underline{\theta})] \sim \chi^2_d$$

ou seja, a função deviance segue uma distribuição Qui-Quadrado com d graus de liberdade, onde d é a dimensão do vetor $\underline{\theta}$.

De acordo com os teoremas apresentados, podemos chegar a algumas das principais propriedades dos estimadores de máxima verossimilhança:

- O estimador de máxima veossimilhança $\hat{\underline{\theta}}$ de $\underline{\theta}$ é assintóticamente não-viciado, isto é, $E(\hat{\underline{\theta}}) \to \underline{\theta}$.
- Assintóticamente $V(\hat{\underline{\theta}}) \to I_E(\underline{\theta})^{-1}$, o qual por uma versão multivariada do limite de Cramér-Rao é o melhor possível, mostrando que o EMV é eficiente para o vetor $\underline{\theta}$.

- Denote $J = I_E(\underline{\theta})^{-1}$, então $V(\hat{\underline{\theta}}) = J$, sendo que, J é uma matriz simétrica e definida positiva, com elementos $J_{i,j} = Cov(\hat{\underline{\theta}}_i, \hat{\underline{\theta}}_j)$ então $J_{i,i}$ é a variância de $\hat{\underline{\theta}}_i$. Denota-se $J_{i,i}^{\frac{1}{2}}$ de desvio padrão de $\hat{\underline{\theta}}_i$.

- Podemos construir intervalos de $100(1-\alpha)\%$ de confiança para θ_i na forma $\hat{\theta}_i \pm Z_{\frac{\alpha}{2}} J_{i,i}^{\frac{1}{2}}$. Intervalos desta forma serão denominados, intervalos de Wald ou baseados em aproximação quadrática da verossimilhança.

- Para regiões de confiança baseados na *deviance* considera-se $[\underline{\theta} \in \Theta : D(\underline{\theta}) < c^*]$, para alguns valores de c^*. Pode-se escolher c^* baseado em justificativas assintoticas de que $D(\underline{\theta}) \sim \chi^2_d$ é uma escolha reazoável para $c^* = c_\alpha$ com $P(\chi^2_d \geq c_\alpha) = \alpha$, por exemplo se $\alpha = 0.05$, então $c_\alpha = 3.84$. Isto gera uma região de $100(1-\alpha)\%$ de confiança. Estes intervalos serão denominados de intervalos deviance.

De acordo com as propriedades apresentadas tem-se duas formas básicas de construir intervalos de confiança. A primeira mais simples é baseada na aproximação quadrática da log-verossimilhança e a segunda olhando diretamente para a função deviance. A segunda opção é bastante complicada computacionalmente, uma vez que ela gera uma equação não linear que precisa ser resolvida numéricamente, o que pode ser custoso. A primeira opção é bastante direta, uma vez obteve-se a matriz de segundas derivadas basta invertê-la e tirar a raiz dos termos da diagonal para se obter o intervalo de confiança para cada parâmetro, marginalmente. Esta abordagem apesar de muito fácil apresenta restrições, por exemplo, não respeita restrições naturais do espaço paramétrico como em parâmetros de variância e correlação pode resultar em limites irreais. Por aproximar sempre por uma forma quadrática os intervalos serão sempre simétricos, o que normalmente não funciona bem para parâmetros de variância e correlação. Em modelos com efeitos aleatórios um interesse natural está em parâmetros de variância, precisão e correlação. Um interesse é testar hipóteses sobre a existência de tais efeitos o que normalmente é feito olhando para as estimativas de variâncias que indexam o modelo, logo esta abordagem é restrita nesta classe mais geral de modelos estatísticos.

É importante deixar claro que a segunda opção resulta em uma região (conjunta) uma elipse, enquanto que pela aproximação é possível obter um intervalo marginal para cada parâmetro, porém baseado em uma aproximação quadrática da superfície de log-verossimilhança. Esta opção é o intuito mais comum, porém não pode ser obtida diretamente apenas com o Teorema 1.6. Por exemplo, suponha que tem-se interesse em um determinado componente do vetor de parâmetros, digamos θ_i baseado na aproximação quadrática podemos facilmente construir um intervalo de confiança, tendo como $\hat{\theta}_L$ e $\hat{\theta}_U$ o seu limite inferior e superior respectivamente. Pelo Teorema 1.6 para o caso em que a dimensão de $\underline{\theta}$ é maior que um, não temos um intervalo desta forma, temos uma elipse o que apesar de mais informativo tem menos apelo prático e apresenta dificuldades de interpretação. Uma forma intuitiva de obter um intervalo da forma $\hat{\theta}_L$ e $\hat{\theta}_U$ é fixar o restante do vetor de parâmetros nas suas estimativas de máxima verossimilhança e olhar em uma direção de cada vez, porém esta abordagem tem uma clara restrição que é não levar em consideração a incerteza associada ao restante do vetor de parâmetros para a construção do intervalo.

Dada a estrutura do problema temos um método simples via aproximação

quadrática, porém não funciona bem quando a superfície de log-verossimilhança é assimétrica. Por outro lado, o método da *deviance* não apresenta esta restrição mais só fornece regiões de confiança, e não limites do tipo $\hat{\theta}_L$ e $\hat{\theta}_U$ para cada parâmetro. Duas abordagens básicas para este problema podem ser consideradas: a primeira é fazer uma reparametrização do modelo, nos parâmetros que apresentam forte assimetria ou são restritos, para torná-los irrestritos e aproximadamente simétricos, obter a variância baseada na aproximação quadrática nesta reparametrização e depois converter para a escala original. Quando este procedimento é satisfatório o custo computacional é pequeno.

Para formalizar esta situação, considere o problema de obter a estimativa pontual e intervalar para um parâmetro de interesse $\phi = g(\underline{\theta})$, onde $g(.)$ é uma função. Desde que $L(\phi) = L(g(\underline{\theta}))$, a função de verossimilhança para ϕ é obtida da função de verossimilhança de $\underline{\theta}$ por uma transformação de escala. Consequentemente, $\hat{\phi} = g(\hat{\underline{\theta}})$, quando o intervalo de confiança digamos $\hat{\theta}_L$ e $\hat{\theta}_U$ for obtido diretamente pela função de verossimilhança, log-verossimilhança ou *deviance*, o intervalo para ϕ pode ser obtido simplesmente transformando os limites obtidos para θ, no caso unidimensional. Esta propriedade é conhecida como invariância do estimador de máxima verossimilhança. Porém, quando o intervalo for obtido pela aproximação quadrática isso não é válido e um Teorema adicional é necessário para esta transformação.

Teorema 4. *Considere obter um intervalo de confiança para $\phi = g(\underline{\theta})$ por invariância temos que $\hat{\phi} = g(\underline{\theta})$ e a variância de $\hat{\phi}$ é dada por*

$$V(\hat{\phi}) = V(g(\underline{\theta})) = \nabla g(\underline{\theta})^\top I_E(\underline{\theta})^{-1} \nabla g(\underline{\theta})$$

onde

$$\nabla g(\underline{\theta}) = \left(\frac{\partial g(\underline{\theta})}{\partial \theta_1}, \ldots, \frac{\partial g(\underline{\theta})}{\partial \theta_1} \right)^\top$$

Vindo do Teorema 4 é imediato o seguinte Teorema.

Teorema 5. *Para um problema de estimação regular se $\phi = g(\underline{\theta})$ são os verdadeiros valores dos parâmetros, então quando $n \to \infty$ tem-se que*

$$\underline{\hat{\phi}} \sim NM_d(\phi, \nabla g(\underline{\theta})^\top I_E(\underline{\theta})^{-1} \nabla g(\underline{\theta}))$$

Pelo Teorema 5, podemos construir intervalos de confiança da mesma forma anterior, porém usando a nova matriz de variância e covariância ponderada pelo gradiente da função $g(.)$, e assim passar de uma reparametrização para outra torna-se uma tarefa trivial. Apesar deste procedimento ser bastante útil, nem sempre é fácil encontrar uma transformação $g(.)$ que torne a log-verossimilhança simétrica. A forma mais efetiva de construir intervalos de confiança para parâmetros de dificil estimação é o intervalo baseado em perfil de verossimilhança.

Seja $\underline{\theta} = (\underline{\phi}^\top, \underline{\lambda}^\top)^\top$, o vetor de parâmetros particionado nos vetores $\underline{\phi}$ e $\underline{\lambda}$, vamos chamar a primeira componente de interesse e a segunda de incômodo, no sentido que desejamos intervalos ou regiões de confiança para $\underline{\phi}$, que pode ser apenas um escalar. Seja $L(\underline{\phi}, \underline{\lambda})$ a verossimilhança para $\underline{\phi}$ e $\underline{\lambda}$. Denota-se $\hat{\underline{\lambda}}_\phi$ a estimativa de máxima verossimilhança de $\underline{\lambda}$ para dado valor de $\underline{\phi}$.

Definição 16 (Verossimilhança perfilada)**.** *A verossimilhança perfilada de* $\underline{\phi}$ *é definida por*

$$\tilde{L}(\underline{\phi}) = L(\underline{\phi}, \hat{\underline{\lambda}}_{\phi})$$

A forma apresentada na definição 16 sugere um procedimento de maximização em duas etapas. A primeira consiste em achar $\hat{\underline{\lambda}}_{\phi}$ que maximiza $l(\underline{\phi}, \underline{\lambda}) = logL(\underline{\phi}, \underline{\lambda})$ com respeito a $\underline{\lambda}$ supondo $\underline{\phi}$ fixo. A segunda maximização busca o maximizar $\tilde{l}(\underline{\phi})$. Assim, uma região ou intervalo de confiança para $\underline{\phi}$ pode ser obtida usando que

$$\tilde{D}(\underline{\phi}) = 2[\tilde{l}(\hat{\underline{\phi}}) - \tilde{l}(\underline{\phi})] \sim \chi^2_d$$

onde d é a dimensão de $\underline{\phi}$. Note que esta forma de construção não impõe nenhum tipo de aproximação, ela pode resultar em intervalos muito assimétricos. Porém, é altamente cara computacionalmente, uma vez que precisamos resolver numericamente uma equação não-linear que para cada avaliação necessita de um algoritmo numérico de maximização.

1.4 Testes de hipóteses

Definição 17. *Chamamos de hipótese estatística qualquer afirmação acerca da distribuição de probabilidades de uma ou mais variáveis aleatórias.*

Definição 18. *Chamamos de teste de uma hipótese estatística a função de decisão* $\chi \to \{a_0, a_1\}$*, em que* a_0 *corresponde à ação de considerar a hipóteses* H_0*, como verdadeira e* a_1 *corresponde à ação de considerar a hipóteses* H_1 *como verdadeira.*

Na definição acima, χ denota o espaço amostral associado à amostra $Y_1, Y_2, \ldots, Y_n$. A função de decisão d divide o espaço amostral χ em dois conjuntos,

$$A_0 = \{(y_1, \ldots, y_n \in \chi; d(y_1, \ldots, y_n) = a_0\}$$

e

$$A_1 = \{(y_1, \ldots, y_n \in \chi; d(y_1, \ldots, y_n) = a_1\}$$

onde $A_0 \cup A_1 = \chi$ e $A_0 \cap A_1 = \emptyset$. Como em A_0 temos os pontos amostrais que levam à aceitação de H_0, vamos chamar de A_0 de região de aceitação e, por analogia, A_1 de região de rejeição de H_0, também chamada de região crítica.

Um teste de hipótese pode resultar em um de dois tipos de erros. Tradicionalmente, esses dois tipos de erros recebem os nomes de erro Tipo I (α) e erro tipo II (β). O erro tipo I ocorre quando rejeitamos H_0 esta é verdadeira. O erro tipo II ocorre quando aceitamos H_0 e esta é falsa. Em termos de probabilidade temos,

$\alpha = P(Y \in A_1|\theta_0)$ e $\beta = P(Y \in A_0|\theta_1)$.

Definição 19. *O poder do teste com região critica* A_1 *para testar* $H_0 : \theta = \theta_0$ *contra* $H_1 : \theta = \theta_1$ *é dado por*

$$\pi(\theta_1) = P(Y \in A_1|\theta_1)$$

Note que $\pi(\theta_1) = 1 - \beta$, onde β é a probabilidade de se cometer o erro do tipo II.

1.4.1 Teste da razão de verossimilhança

Definição 20. *A estatística do teste da razão de verossimilhança para testar* $H_0 : \theta \in \Theta_0$ *versus* $H_1 : \theta \in \Theta_0^c$ *é*

$$\lambda(\underline{x}) = \frac{sup_{\Theta_0} L(\theta|\underline{y})}{sup_{\Theta} L(\theta|\underline{y})}$$

O teste da razão de verossimilhança (TRV) é qualquer teste que tenha uma região de rejeição da forma $\underline{y} : \lambda(\underline{y}) \leq c$ *onde* c *é qualquer número que satisfaça* $0 \leq c \leq 1$.

Para testar $H_0 : \theta = \theta_0$ versus $H_1 : \theta \neq \theta_)$, suponha $Y_1, \ldots, Y_n$ sejam iid $f(\underline{y}|\theta)$, $\hat{\theta}$ seja o EMV de θ, e $f(\underline{y}|\theta)$ satisfaça as condições de regularidade. Desse modo, de acordo com H_0, pelo Teorema 1.6 à medida que $n \to \infty$

$$-2 \log \lambda(\underline{x}) \to \chi_1^2.$$

1.4.2 Testes de Wald

Suponha que queiramos testar a hipótese bilateral $H_0 : \theta = \theta_0$ versus $H_1 : \theta \neq \theta_0$. Um teste aproximado poderia ter como base a estatística $Z_n = (W_n - \theta_0)/S_n$ e rejeitaria H_0 se, e somente se, $Z_n < -z_{\alpha/2}$. Se H_0 for verdadeira, então $\theta = \theta_0$ e Z_n converge em distribuição para $Z \sim N(0,1)$. Portanto, a probabilidade do Erro Tipo I, $P_{\theta_0}(Z_n < -z_{\alpha/2} \quad \text{ou} \quad Z_n > z_{\alpha/2}) \to P(Z < -z_{\alpha/2} \quad \text{ou} \quad Z > z_{\alpha/2}) = \alpha$, este é, assintóticamente, um teste de tamanho α. Em geral, um teste de Wald é um teste com base em uma estatística da forma,

$$Z_n = \frac{W_n - \theta_0}{S_n}$$

onde θ_0 é um valor hipotético do parâmetro θ, W_n é um estimador de θ e S_n é um erro padrão de W_n, uma estimativa do desvio padrão de W_n. Se W_n for o EMV para θ, então, $\frac{1}{\sqrt{I_O(\hat{\theta})}}$ é um erro padrão razoável para W_n.

1.4.3 Teste escore

Definição 21. *A estatística de escore é definida como*

$$U(\theta) = \frac{\partial}{\partial \theta} l(\theta|\underline{Y})$$

Sabemos que para todo θ, $E_\theta(U(\theta)) = 0$. Em particular, se estivermos testando $H_0 : \theta = \theta_0$ e se H_0 for verdadeira, então $U(\theta)$ tem média 0. Além disso,

$$V_\theta(U(\theta)) = -E_\theta \left(\frac{\partial^2}{\partial y^2} l(\theta|\underline{Y}) \right) = I_E(\theta)$$

o número de informações é a variância da estatística escore. A estatística de teste para o teste de escore é

$$Z_S = U(\theta_0)/\sqrt{I_E(\theta_0)}.$$

Se H_0 for verdadeira, Z_S tem média 0 e variância 1.

1.5 Exemplo 1 - Distribuição Poisson

Seja $Y_i \sim P(\lambda)$ com $i = 1, \ldots, n$, variáveis aleatórios independentes. A função de verossimilhança é o produto das n distribuições de Poisson com mesmo parâmetro λ. Então

$$L(\lambda) = \prod_{i=1}^{n} \frac{\exp^{-\lambda} \lambda^{y_i}}{y_i!},$$

é a verossimilhança deste modelo. De forma equivalente podemos trabalhar com a função de verossimilhança relativa. Sendo, $\hat{\lambda}$ o EMV para λ a verossimilhança relativa é dada por $LR(\lambda) = \frac{L(\lambda)}{L(\hat{\lambda})}$. Que apresenta a propriedade de estar sempre no intervalo unitário o que facilita a construção e visualização de gráficos.

Outra possibilidade é usar a função de log-verossimilhança $l(\lambda) = \log L(\lambda)$ que normalmente é preferida para se trabalhar analitica e computacionalmente, uma vez, que esta função normalmente é mais fácil de derivar que a $L(\lambda)$. E por fim, podemos ainda trabalhar com a função *deviance* dada por, $D(\lambda) = 2\{l(\hat{\lambda}) - l(\lambda)\}$, que em geral é preferida para construção de intervalos de confiança, devida a suas propriedades assintóticas. A Figura 1.10, apresenta as quatro formas básicas de visualizar a função de verossimilhança.

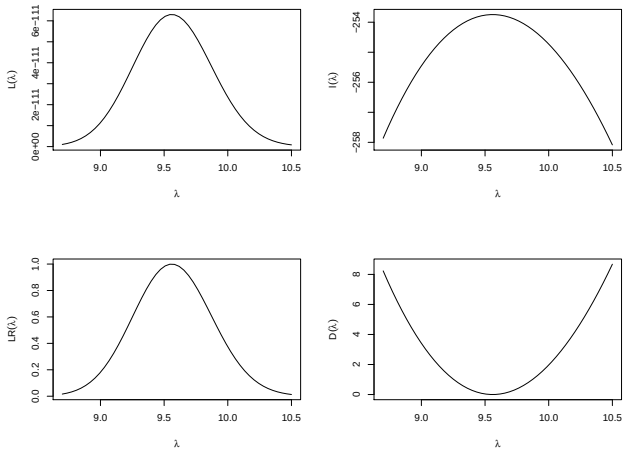

Figura 1.1: Diferentes formas de visualizar a função de verossimilhança.

Apesar das quatro formas serem equivalentes a forma mais conveniente para encontrar o estimador de máxima verossimilhança é a log-verossimilhança. Note que usando a verossimilhança fica bastante complicado encontrar o máximo da função usando técnicas convencionais de cálculo, usando a função *deviance* é necessário encontrar a raiz, ou seja, onde a função toca o eixo x, o que mesmo na situação extremamente simples colocada aqui, leva a uma função não-linear e métodos numéricos são necessários para a solução do problema. Apenas para exemplificar vamos encontrar o estimador de máxima verossimilhança para λ, analiticamente usando a log-verossimilhança. Partindo da verossimilhança temos,

$$
\begin{aligned}
L(\lambda) &= \prod_{i=1}^{n} \frac{\exp^{-\lambda} \lambda^{Y_i}}{Y_i!} \\
l(\lambda) &= -n\lambda + \log(\lambda) \sum_{i=1}^{n} Y_i - \sum_{i=1}^{n} \log Y_i! \\
U(\lambda) &= -n + \frac{1}{\lambda} \sum_{i=1}^{n} Y_i \\
\hat{\lambda} &= \frac{\sum_{i=1}^{n} Y_i}{n} \\
I_O(\lambda) &= \frac{\sum_{i=1}^{n} Y_i}{\lambda^2} \\
V(\hat{\lambda}) &= I_O(\hat{\lambda})^{-1}
\end{aligned}
$$

Como o estimador de máxima verossimilhança é uma função de uma variável aleatória ele também é uma variável aleatória. Conforme as propriedades apresentadas o EMV é assintóticamente não viciado e sua distribuição amostral é assintóticamente Gaussiana. Para exemplificar estas propriedades vamos fazer um pequeno estudo de simulação, para verificar como se comporta o viés e a distribuição do EMV conforme o tamanho da amostra aumenta.

Para isto, vamos simular 1000 conjunto de dados de acordo com o modelo Poisson com $\lambda = 10$. Vamos retirar amostras de tamanho $5, 50, 100$ e 1000, em cada amostra calcular o EMV. A Figura 1.2 apresenta os resultados deste estudo de simulação. Sabemos que pelas propriedades do EMV apresentados temos que $\hat{\lambda} \sim N(\lambda, \frac{\lambda^2}{\sum_{i=1}^{n} y_i})$, sobrepondo a distribuição assintótica teórica a distribuição empirica obtida pela simulação chegamos a Figura 1.2.

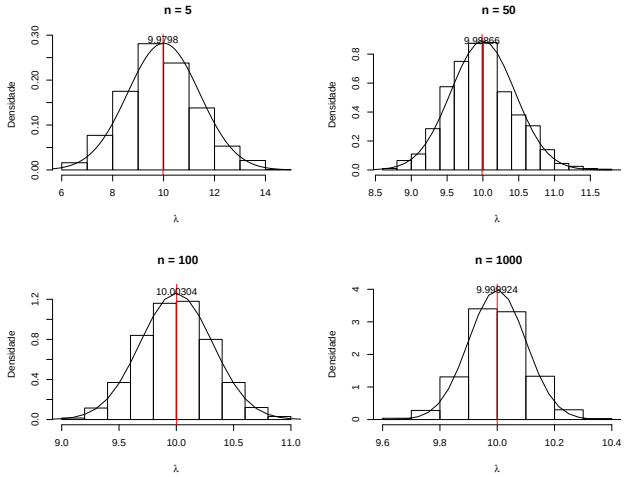

Figura 1.2: Viés e distribuição assintótica estimador de máxima verossimilhança modelo de Poisson.

Como é possível visualizar na Figura **??** a distribuição empírica apresenta um comportamento muito próximo da distribuição teórica, mesmo para amostras pequenas $n = 5$ e $n = 50$, o viés vai diminuindo conforme a amostra aumenta. É também evidente que com uma amostra maior a variância do EMV vai diminuindo, até no caso limite quando $n \to \infty$ atinge o 0 mostrando a consistência do EMV.É interessante observar que mesmo com uma amostra pequena, os resultados válidos assintóticamente já apresentam resultados excelentes. É claro que este é um exemplo simples, porém como veremos mesmo em modelos mais complexos, como nos modelos lineares generalizados (GLM) estes resultados permanecem igualmente bons.

1.6 Exemplo 2 - Intervalos de confiança

Como vimos na Seção 1.2 temos três formas básicas para construir intervalos baseados em verossimilhança. A mais simples é baseada na verossimilhança relativa, porém o intervalo não tem interpretações probabilística. A segunda baseada na função *Deviance* e a terceira usando uma aproximação quadrática da superfície de log-verossimilhança. As últimas duas opções trazem a interpretação frequentista para o intervalo de confiança. Desta forma, os objetivos deste exemplo são: mostrar como construir intervalos de confiança pelas três abordagem, explorar as relações existentes e discutir as dificuldades computacionais que ocorrem quando trabalhamos diretamente com a função *Deviance*.

Para isto, considere o caso onde amostras independentes de uma variável aleátoria Y_i com $i = 1, 2, \ldots, n$ cada uma com distribuição exponencial de parâmetro θ esteja disponível, vamos denotar isto por, $Y_i \sim \exp(\theta)$. O objetivo é estimar o parâmetro θ e um intervalo de confiança, digamos $\hat{\theta}_L$ e $\hat{\theta}_U$ que tenha probabilidade $1 - \alpha$ de conter θ. Note que a estrutura probabilística está em $\hat{\theta}_L$ e $\hat{\theta}_U$.

O primeiro passo é sempre escrever a função de verossimilhança,

$$L(\theta) = \prod_{i=1}^{n} \theta \exp^{-\theta y_i} = \theta^n \exp^{-\theta \sum_{i=1}^{n} y_i}$$

Consequentemente, a função de log-verossimilhança

$$l(\theta) = n \log \theta - \theta \sum_{i=1}^{n} y_i = n \log \theta - \theta n \overline{y}$$

Derivando a log-verossimilhança em θ chegamos a função escore

$$U(\theta) = n\theta^{-1} - n\overline{y}$$

Para encontrar a estimativa de máxima verossimilhança basta igualar a função escore a 0 e isolar o θ isto resulta

$$\hat{\theta} = \frac{1}{\overline{y}}$$

A segunda derivada da função de log-verossimilhança tem um papel fundamental na construção de intervalos de confiança, uma vez que ela mede a

curvatura da função em torno do ponto de máximo. Em termos estatístico trabalhamos com o seu recíproco, a informação observado/esperado, uma vez esta quantidade descreve o comportamento assintótico dos estimadores de máxima verossimilhança, já que sua inversa é a variância do estimador. No caso deste exemplo,

$$I_O(\hat{\theta}) = n\hat{\theta}^{-2}$$

Para encontrar o intervalo de confiança vamos começar pela função *Deviance*, lembre-se que de acordo com o Teorema , a *Deviance* segue uma distribuição Qui-quadrado com d graus de liberdade, neste caso $d = 1$. Dado isto, podemos escolher o valor de c^* olhando para os quantis da distribuição qui-quadrado com 1 grau de liberdade e $1 - \alpha\%$ de confiança. Neste exemplo, vamos fizar $\alpha = 0.05\%$ o que leva a um valor de aproximadamente $c^* = 3.84$. Com isto, temos todos os ingredientes para construir o intervalo de confiança baseado na função *Deviance*.

$$\begin{aligned} D(\theta) &= 2[l(\hat{\theta}) - l(\theta)] \leq c* \\ &= 2[n\log\hat{\theta} - \hat{\theta}n\overline{y} - [n\log\theta - \theta n\overline{y}]] \leq c^* \\ &= 2n[\log\left(\frac{\hat{\theta}}{\theta}\right) + \overline{y}(\theta - \hat{\theta})] \leq c^* \end{aligned} \tag{1.1}$$

Como é possível ver na equação mesmo numa situação simples como a deste exemplo a *Deviance* necessita da solução de um sistema não-linear que não tem solução analítica. Desta forma, algum método numérico para encontrar raízes de funções não-lineares é necessário.

Uma outra opção para encontrar o intervalo de confiança é fazer uma aproximação em Séries de Taylor para a função $l(\theta)$ em torno do ponto $\hat{\theta}$ e usar esta aproximação para obter o intervalo de confiança. Isto é equivalente a usar o resultado do Teorema , ou seja, os intervalos são da forma

$$\hat{\theta} \pm Z_{\frac{\alpha}{2}}\sqrt{V(\hat{\theta})}$$

No caso da distribuição exponencial, temos que a $V(\hat{\theta}) = I_O(\hat{\theta})^{-1} = \frac{\hat{\theta}^2}{n}$, logo o intervalo de confiança é dado por

$$\hat{\theta}_L = \hat{\theta} - Z_{\frac{\alpha}{2}}\frac{\hat{\theta}}{/}\sqrt{n} \quad \text{e} \quad \hat{\theta}_L = \hat{\theta} - Z_{\frac{\alpha}{2}}\frac{\hat{\theta}}{/}\sqrt{n}$$

Note que neste caso a construção do intervalo de confiança fica bastante facilitada. Conhecendo o valor de $Z_{\frac{\alpha}{2}}$ o intervalo se resume a uma conta simples. Como a distribuição amostral de $\hat{\theta}$ é assintóticamente gaussiano podemos escolher o valor de $Z_{\frac{\alpha}{2}}$ como o quantil da distribuição Normal, com $1 - \alpha$ % de confiança, escolhendo $\alpha = 0.05$ % , temos aproximadamente que $Z_{\frac{\alpha}{2}} = 1.96$.

A última opção que vamos considerar é a construção do intervalo olhando para a verossimilhança relativa. Lembre-se que um intervalo baseado em verossimilhança é da forma $LR(\theta) = \frac{L(\theta)}{L(\hat{\theta})} \geq r$, onde r representa a perda percentual em relação ao máximo da função de verossimilhança, para o qual admite-se ainda

obter uma boa estimativa de θ. Para escolher um valor de r compatível com o escolhido para a *deviance*, note o seguinte:

$$\begin{aligned} \frac{L(\theta)}{L(\hat{\theta})} &\geq r \\ \log\left[\frac{L(\theta)}{L(\hat{\theta})}\right] &\geq \log r \\ 2[l(\theta) - l(\hat{\theta})] &\geq 2 * \log r \\ -3.84 &\geq 2 * \log r \\ r &\approx 0.146 \end{aligned}$$

Com isso, podemos observar que olhar para a verossimilhança relativa ou para a *deviance* são formas equivalentes de construir o intervalo de confiança. A vantagem em usar a *deviance* é que conhecemos a sua distribuição assintótica o que permite a construção de intervalos de confiança com a desejada estrutura probabilística. Da mesma forma que para a *deviance* olhando para a verossimilhança também precisamos resolver um sistema não-linear que novamente será necessário o uso de algum método numérico para encontrar as raízes da equação $LR(\theta) \geq r$.

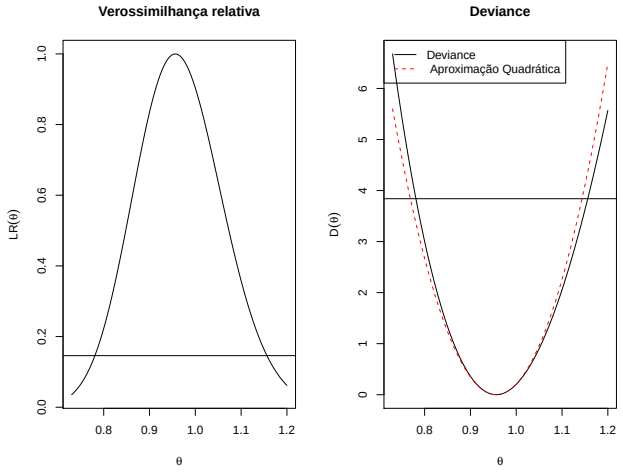

Figura 1.3: Verossimilhança relativa e função deviance exata e aproximada.

Como é possível ver na Figura a função *deviance* é razoavelmente aproximada por uma forma quadrática. Para obter os limites inferior e superior do intervalo de confiança pela aproximação quadrática temos:

$$\hat{\theta}_L = \hat{\theta} - Z_{\frac{\alpha}{2}}\sqrt{\frac{\theta^2}{n}} \quad \text{e} \quad \hat{\theta}_U = \hat{\theta} - Z_{\frac{\alpha}{2}}\sqrt{\frac{\theta^2}{n}}$$

Para a amostra realizada utilizada para gerar os gráficos temos os seguintes valores:

$$\hat{\theta}_L = 0.956 - 1.96\sqrt{\frac{0.956^2}{100}} \quad \text{e} \quad \hat{\theta}_U = 0.956 + 1.96\sqrt{\frac{0.956^2}{100}}$$

O que resulta no seguinte intervalo de confiança,

$$\hat{\theta}_L = 0.768 \quad \text{e} \quad 1.143.$$

Usando a função *Deviance* precisamos resolver o sistema não-linear. Em R podemos facilmente resolver este problema usando as funções do pacote *rootSolve* [**?**]. O primeiro passo é escrever a função *Deviance*,

```
> deviance <- function(theta, theta.hat, y) {
+     n <- length(y)
+     dv <- 2 * n * (log(theta.hat/theta) + mean(y) * (theta -
+         theta.hat))
+     return(dv - qchisq(0.95, df = 1))
+ }
```

Uma vez com a função escrita podemos encontrar suas raízes usando a função *uniroot.all()*.

```
> uniroot.all(deviance, interval = c(0, 10), theta.hat = 1/mean(y),
+     y = y)

[1] 0.7808944 1.1561483
```

Conforme a Figura mostrava o intervalo aproximado pela forma quadrática apresenta um leve deslocamente para a esquerda quadno comparado com o intervalo baseado na função *Deviance* diretamente. É importante ter bastante cuidado na interpretação destes intervalos. Lembre-se que de acordo com a interpretação frequentista de probabilidade, se realizarmos por exemplo 100 vezes o mesmo experimento e em cada um destes experimentos calcularmos o respectivo intervalo de confiança esperamos que $1 - \alpha\%$ dos intervalos construídos contenham o verdadeiro valor do parâmetro. Isto pode ser melhor entendido com o seguinte estudo de simulação.

```
> set.seed(12)
> ic <- matrix(NA, ncol = 2, nrow = 100)
> for (i in 1:100) {
+     y <- rexp(100, rate = 1)
+     theta.hat <- 1/mean(y)
+     ic[i, ] <- uniroot.all(deviance, interval = c(0, 10), theta.hat = 1/mean(y),
+         y = y)
+ }
```

No código **??** na linha 3 simulamos a cada passo o laço $for()$ uma nova realização da variável aleatória Y, com esta realização calcular a estimativa de máxima verossimilhança e o seu respectivo intervalo de confiança baseado na *Deviance* e guardamos o resultado em um objeto chamado *ic*. De acordo com a interpretação frequentista dos 100 intervalos de confiança que calculamos esperamos que em 95 deles contenham o verdadeiro valor do parâmetro neste caso $\theta = 1$. O gráfico apresentado na Figura 1.4 ajuda a entender a situação.

Neste caso conforme esperado temos 5 dos intervalos que não contem o valor verdadeiro do parâmetro. É claro que por se tratar de um exemplo de simulação variações podem ocorrer. A taxa de cobertura de um intervalo é a quantidade

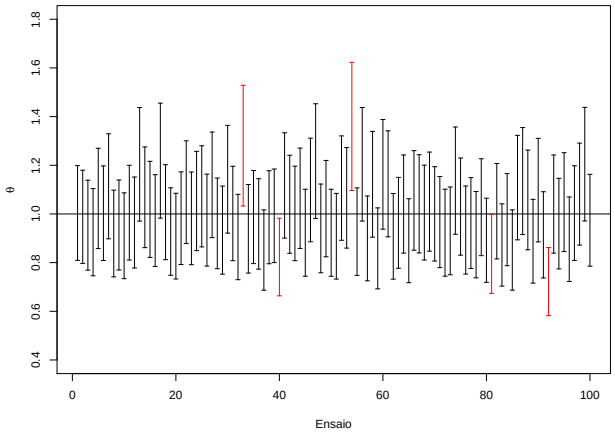

Figura 1.4: Interpretação frequentista de intervalos de confiança.

de vezes em que os intervalos contem o verdadeiro valor do parâmetro sobre o total de ensaios simulados. Neste caso 95/100%, ou seja, a taxa de cobertura é de 95% igual ao nível nominal especifica para este intervalo.

A taxa de cobertura é uma forma muito utilizada para avaliar métodos de construção de intervalos de confiança. Em modelos mais complexos principalmente os que envolvem efeitos aleatórios, os componentes de variância são de dificil estimação e os seus intervalos de confiança principalmente os construídos baseados em aproximação quadrática da log-verossimilhança apresentam taxa de cobertura abaixo do nível nominal especificado.

1.7 Exemplo 3 - Testes de hipóteses

Em situações práticas podemos estar interessados em testar se o parâmetro de um dado modelo é igual, maior ou menor que algum valor de interesse. Conforme descrito na Seção 1.4, um teste de hipóteses é qualquer afirmação acerca da dsitribuição de probabilidade de uma ou mais variáveis aleatórias. Considere a situação onde observamos uma amostra aleatória de tamanho n de uma população com distribuição de Poisson de parâmetro λ. Suponha que o interesse é testar sob a hipótese nula $H_0 : \lambda = \lambda_0$ contra uma hipótese alternativa $H_1 : \lambda \neq \lambda_0$. Vimos na Seção 1.4 três formas de construir testes de hipóteses que podem ser usadas para concluir sobre as hipóteses levantadas. Como exemplo didático, vamos obter as três estatísticas de testes para o caso onde $X_i \sim P(\lambda)$. Antes de construir os testes propriamente dito vamos obter algumas quantidades relevantes que facilitam a cosntrução dos testes de hipóteses.

Como as amostras são independentes a função de verossimilhança deste modelo é dada por,

$$L(\lambda) = \prod_{i=1}^{n} \frac{\exp^{\lambda} \lambda_i^x}{x_i!} = \frac{\exp^{-n\lambda} \lambda^{\sum_{i=1}^{n} x_i}}{\prod_{i=1}^{n} x_i!}.$$

A função de log-verossimilhança,

$$l(\lambda) = -\lambda n + \sum_{i=1}^{n} \log \lambda - \sum_{i=1}^{n} \log x_i!.$$

A função escore é obtida derivando a log-verossimilhança em λ,

$$U(\lambda) = -n + \frac{\sum_{i=1}^{n} x_i}{\lambda}$$

que sendo resolvida fornece a estimativa de máxima verossimilhança

$$\hat{\lambda} = \frac{\sum_{i=1}^{n} x_i}{n} = \overline{x}.$$

Além disso, temos que a informação observada é dada por

$$I_O(\lambda) = \frac{\sum_{i=1}^{n} x_i}{\lambda^2}.$$

Para obter informação esperada avaliamos a esperança da informação observada

$$I_E(\lambda) = E(I_O(\lambda)) = E(\frac{\sum_{i=1}^{n} x_i}{\lambda^2}) = \frac{n}{\lambda}.$$

Vamos começar pelo teste de razão de verossimilhança. A estatística do teste de razão de verossimilhança neste caso toma a seguinte forma:

$$\lambda(\underline{x}) = \frac{L(\lambda_0|\underline{x})}{L(\hat{\lambda}|\underline{x})}$$

Que é a verossimilhança relativa, note que $-2\log\lambda(\underline{x})$ é exatamente a função *Deviance* usada para construir intervalos de confiança. Então,

$$-2\log\lambda(\underline{x}) = -2\log\left(\frac{\exp^{-n\lambda_0}\lambda_0^{\sum_{i=1}^{n} x_i}}{\exp^{-n\hat{\lambda}}\hat{\lambda}^{\sum_{i=1}^{n} x_i}}\right) = 2n\left[(\lambda_0 - \hat{\lambda}) - \hat{\lambda}\ log\left(\frac{\lambda_0}{\hat{\lambda}}\right)\right]$$

A hipótese nula $H_0 : \lambda = \lambda_0$ será rejeitada se, e somente se, $-2\log\lambda(\underline{x}) \geq \chi^2_{1,\alpha}$. A probabilidade de Erro do Tipo I será aproximadamente α.

O segundo método para construção de testes de hipóteses é o método de Wald. Se $\hat{\lambda}$ é o estimador de máxima verossimilhança a estatística do teste de Wald é dada por

$$T_w = \frac{\hat{\lambda} - \lambda}{\sqrt{V(\hat{\lambda})}} \sim N(0,1)$$

Sabemos que $V(\hat{\lambda}) = I_E(\hat{\lambda})^{-1}$, então $V(\hat{\lambda}) = \frac{\hat{\lambda}}{n}$. Logo o teste de Wald no caso Poisson resume-se a

$$T_w = \frac{\overline{x} - \theta}{\sqrt{\frac{\overline{x}}{n}}} \sim N(0,1)$$

Dado a distribuição assintótica do teste para encontrar a região de rejeição basta encontrar o quantil da distribuição Normal padrão correspondente ao nível de confiança desejado.

A terceira opção é a estatística de teste *escore* que é dada por,

$$T_E = \frac{U(\lambda_0}{\sqrt{I_E(\lambda_0)}}$$

substituindo as quantidades previamente encontradas temos para o caso Poisson,

$$T_E = \frac{-n + \sum_{i=1}^{n} x_i/\lambda_0}{\sqrt{n/\lambda_0}} \sim N(0,1).$$

A Figura **??** ilustra a construção dos testes hipóteses e os relaciona através da função de verossimilhança do modelo.

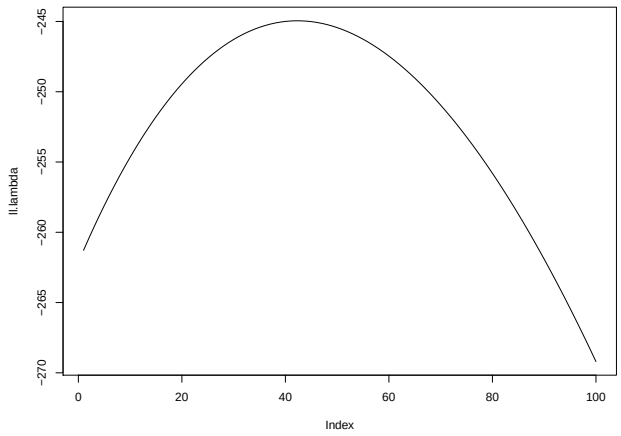

Figura 1.5: Diferentes formas de visualizar a função de verossimilhança.

Para exemplificar a execusão do teste de hipóteses vamos utilizar um conjunto de dados simulados com $n = 100$ amostras aleatórias de uma Poisson com $\lambda = 10$. Vamos supor que o interesse seja testar sob $H_0 : \lambda = 8$ contra $H_1 : \lambda \neq 8$. As funções abaixo montam a estrutura do teste de hipótese de acordo com cada abordagem.

```
> trv <- function(Est, H0, alpha, ...) {
+     critico <- qchisq(1 - alpha, df = 1)
+     est.calc <- Est(H0, ...)
+     print(ifelse(est.calc < critico, "Aceita H0", "Rejeita H0"))
+     return(c(est.calc, critico))
+ }
> wald <- function(H0, EMV, V.EMV, alpha) {
+     critico <- qnorm(1 - alpha/2)
+     Tw <- (EMV - H0)/sqrt(V.EMV)
+     print(ifelse(Tw < critico, "Aceita H0", "Rejeita H0"))
+     return(c(Tw, critico))
```

```
+ }
> escore <- function(H0, U, Ie, alpha, ...) {
+     critico <- qnorm(1 - alpha/2)
+     Te <- U(H0, ...)/sqrt(Ie(H0, ...))
+     print(ifelse(Te < critico, "Aceita H0", "Rejeita H0"))
+     return(c(Te, critico))
+ }
```

Usando as funções baseados na amostra simulada temos

```
> set.seed(123)
> x <- rpois(100, lambda = 10)
> Est <- function(H0, x) {
+     n <- length(x)
+     EMV <- mean(x)
+     lv <- 2 * n * ((H0 - EMV) + EMV * log(EMV/H0))
+     return(lv)
+ }
> trv(Est = Est, H0 = 8, alpha = 0.05, x = x)

[1] "Rejeita H0"
[1] 32.660809  3.841459

> wald(H0 = 8, EMV = mean(x), V.EMV = mean(x)/length(x), alpha = 0.05)

[1] "Rejeita H0"
[1] 5.370358 1.959964
```

O teste escore é ligeiramente mais complicado uma vez que é necessário programar a função escore e uma função para avaliar a informação esperada.

```
> fc.escore <- function(lambda, x) {
+     n <- length(x)
+     esco <- -n + sum(x)/lambda
+     return(esco)
+ }
> Ie <- function(lambda, x) {
+     n <- length(x)
+     I <- n/lambda
+     return(I)
+ }
> escore(H0 = 8, U = fc.escore, Ie = Ie, alpha = 0.05, x = x)

[1] "Rejeita H0"
[1] 5.904342 1.959964
```

Neste caso os três testes levam a mesma conclusão. Em geral, o teste escore é o mais complicado de se obter, uma vez que precisa da função escore e da informação esperada ou observada. Apesar destas quantidades poderem ser obtidas numericamente em geral o esforço computacional é maior que pelas outras duas abordagens. O teste de Wald é o mais utilizado pelo menos de forma

inicial uma vez que sua construção é simples. O teste de razão de verossimilhança é também muito utilizado, tanto para testar valores para um determinado parâmetro quanto para a escolha de modelos estatísticos. Na situação em que usamos métodos numéricos para maximização da função de log-verossimilhança um subproduto é a informação observada que pode ser usada diretamente na estatística de teste de Wald. Além disso, quando comparamos modelos aninhados a diferença entre os valores da log-verossimilhança dos modelos, permite a construção do testa da razão de verossimilhança de forma bastante direta, porém requer duas otimizações, uma para cada modelo, enquanto que o de Wald apenas uma.

1.8 Exemplo 4 - Reparametrização

Em diversas aplicações pode ser de interesse alguma função de um parâmetro do modelo, ou pode ser mais simples estimar em uma certa parametrização do que em outra. Por exemplo, no caso de parâmetros de variância em geral é mais estável numéricamente estimar a sua raiz ou o log da raiz. Note também que reparametrizações mudam o intervalo de busca em algoritmos de maximização. Novamente em parâmetros de variância digamos σ^2 tem como seu intervalo de busca o $\Re^+$, enquanto que se ao invés de estimar σ^2 estimarmos um $\phi = \log\sqrt(\sigma^2)$ o intervalo de busca será toda a reta real, o que normalmente é mais conveniente quando trabalha-se com algoritmos de maximização numérica.

Como exemplo didático deste problema, seja $X_i : i = 1, \ldots, n$ variáveis aleatórias independentes com função densidade de probabilidade dada por:

$$f(x;\theta) = 2\theta x \exp^{-\theta x^2} \quad : \quad x \leq 0$$

Considere que seja de interesse a reparametrização $\theta = \frac{1}{\mu}$. Vamos primeiro fazer todo o processo de inferencia considerando que queremos estimar o parâmetro θ na sequencia consideramos todo o processo considerando o parâmetro μ para finalizar vamos mostrar como passar de uma reparametrização para outra, ou seja, com as estimativas de θ obter as estimativas tanto pontuais quanto intervalares para μ. Começamos escrevendo a função de verossimilhança,

$$L(\theta) = \prod_{i=1}^{n} 2\theta x_i \exp^{-\theta x_i^2} = (2\theta)^n \prod_{i=1}^{n} \log x_i \exp^{-\theta \sum_{i=1}^{n} x_i}$$

Logo a função de log-verossimilhança,

$$l(\theta) = n\log 2 + n\log\theta + \sum_{i=1}^{n} \log x_i - \theta \sum_{i=1}^{n} x_i^2.$$

Derivando em relação a θ chegamos a função escore,

$$U(\theta) = \frac{\theta}{n} - \sum_{i=1}^{n} x_i^2$$

Igualando a zero encontramos a estimativa de máxima verossimilhança,

$$\hat{\theta} = \frac{n}{\sum_{i=1}^{n} x_i^2}$$

Para a construção do intervalo de confiança usando a aproximação quadrática precisamos da segunda derivada, da qual derivamos a informação observada e/ou esperada, neste caso temos,

$$I_O(\theta) = -\frac{\partial l(\theta)}{\partial \theta} = \frac{n}{\theta^2}$$

Para obter a informação esperada basta obter a esperança da informação observada,

$$I_E(\theta) = E(I_O(\theta)) = E\left(\frac{n}{\theta^2}\right) = \frac{n}{\theta^2}$$

Note que neste caso a $I_O(\theta)$ e a $I_E(\theta)$ coincidem, porém isso não é em geral válido, apenas em alguns poucos casos particulares. Com os cálculos desenvolvidos acima podemos estimar o parâmetro θ e encontrar um intervalo de confiança aproximado usando a aproximação quadrática. Para o caso de desejarmos obter intervalos de confiança baseado na função *Deviance* precisamos resolver a seguinte equação não-linear,

$$D(\theta) = 2\left[n\log\left(\frac{\hat{\theta}}{\theta}\right) + (\hat{\theta} - \theta)\sum_{i=1}^{n} x_i^2\right] \leq c^*$$

Isto pode ser resolvido usando o método de Newton-Raphson conforme será explicado na sequência. Por agora, podemos resolver conforme no Exemplo 2, usando a função *uniroot.all*(), que implementa o Newton-Raphson para uma função qualquer. Com isso, temos todo o processo de inferência completo, podemos estimar pontualmente, obter intervalo de confiança aproximado ou baseado na *Deviance*.

Mas não estamos interessados em θ, mas sim em μ, então começamos escrevendo a verossimilhança para μ.

$$L(\mu) = \prod_{i=1}^{n} 2\mu^{-1} x_i \exp^{-\frac{x_i^2}{\mu}} = (2\mu^{-1})^n \exp^{-\frac{1}{\mu}\sum_{i=1}^{n}} \prod_{i=1} \log x_i.$$

A log-verossimilhança é dada por,

$$l(\mu) = n\log 2 - n\log\mu - \mu^{-1}\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} \log x_i.$$

Derivando em μ obtemos a função escore,

$$U(\mu) = -\frac{n}{\mu} + \mu^{-2}\sum_{i=1}^{n} x_i^2.$$

Igualando a zero chegamos a estimativa de máxima verossimilhança,

$$\hat{\mu} = \frac{\sum_{i=1}^{n} x_i^2}{n}$$

Para a informação observada tem-se

$$\begin{aligned} I_O(\mu) &= -\frac{\partial^2 l(\mu)}{\partial \mu} = -\frac{\partial}{\partial \mu} - \frac{n}{\mu} + \mu^{-2} \sum_{i=1}^{n} x_i^2 \\ &= n\mu^{-2} - 2\mu^{-3} \sum_{i=1}^{n} x_i^2 \end{aligned}$$

Para obter a informação esperado precisamos tomar a esperança de $I_O(\mu)$,

$$I_E(\mu) = E(I_O(\mu)) = E\left(n\mu^{-2} - 2\mu^{-3} \sum_{i=1}^{n} X_i^2 \right)$$

Note a dificuldade que temos neste passo, uma vez que é necessário calcular a $E(X_i^2)$ que é a solução da integral $E(X_i^2) = \int_0^\infty X_i^2 2\mu^{-1} X_i \exp^{-\frac{X_i^2}{\mu}} dx = 2\mu$ que neste caso é possível de ser obtida analiticamente. Substituindo na equação acima temos,

$$I_E(\mu) = E(-n\mu^{-2} + 2\mu^{-3} n 2\mu) = n\mu^{-2}$$

Note que neste caso a informação observada é diferente da esperada e que isto afeta a construção do intervalo de confiança baseado na aproximação quadrática, uma vez que muda a estimativa de variância do estimador de máxima verossimilhança. As duas formas são equivalentes assintóticamente, porém na situação real temos apenas uma amostra finita observada. Na maioria das situações em modelagem estatística quando métodos numéricos são utilizados não temos a opção de escolher entre a informação observada e esperada, quando a segunda derivada é obtida numricamente estamos diretamente usando a informação observada. Podemos dizer que usar a informação observada é acreditar plenamente na amostra observada, enquanto usar a informação esperada estamos emprestando mais informação do modelo. Pensamos na informação observada como uma medida de curvatura local, por outro lado a informação esperada é uma medida de curvatura global.

A construção de intervalos de confiança baseados diretamente da função *deviance* é feita resolvendo a seguinte equação não linear,

$$D(\mu) = 2 \left[n \log\left(\frac{\mu}{\hat{\mu}}\right) + (\mu^{-1} - \hat{\mu}^{-1}) \sum_{i=1}^{n} x_i^2 \right]$$

Para exemplificar considere que a seguinte amostra foi observada $y_i = 0.19; 1.68; 2.81; 0.59; 1.18$. Vamos fazer um gráfico da verossimilhança em cada parametrização. Começamos escrevendo as funções em R.

```
> L.theta <- function(theta, y) {
+     n <- length(y)
+     saida <- (2 * theta)^n * prod(y) * exp(-theta * sum(y^2))
+     return(saida)
+ }
> L.mu <- function(mu, y) {
+     n <- length(y)
+     saida <- (2 * mu^-1)^n * prod(y) * exp(-(1/mu) * sum(y^2))
```

```
+     return(saida)
+ }
```

Vamos entrar com os dados no R em forma de vetor e calcular as estimativas de máxima verossimilhança para θ e μ.

```
> y <- c(0.19, 1.68, 2.81, 0.59, 1.18, 0.19, 1.68, 2.81, 0.59,
+     1.18)
> theta <- length(y)/sum(y^2)
> mu <- sum(y^2)/length(y)
> c(theta, mu)

[1] 0.4001569 2.4990200
```

Note que pela propriedade de invariância você pode obter μ partindo de θ e vice-versa. Os gráficos da função de verossimilhança e log-verossimilhança nas duas parametrizações são apresentados na Figura **??**.

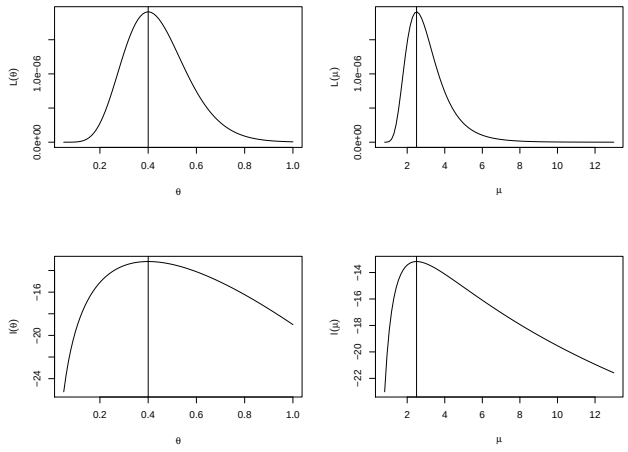

Figura 1.6: Diferentes formas de visualizar a função de verossimilhança.

Para a construção dos intervalos de confiança baseados na *Deviance* basta resolver as respectivas equações não lineares. Além disso, podemos obter os intervalos de confiança aproximados usando a informação observada e/ou esperada, dependendo da situação. A Figura 1.7, apresenta as funções *deviance* exata e aproximada em cada parametrização. Para isto, precisamos criar cada uma das funções, para depois poder avalia-las.

```
> dev.theta <- function(theta, theta.hat, y, desloca = 0) {
+     n <- length(y)
+     saida <- 2 * (n * log(theta.hat/theta) - (theta.hat - theta) *
+         sum(y^2))
+     return(saida - desloca)
+ }
> dev.mu <- function(mu, mu.hat, y, desloca = 0) {
```

```
+     n <- length(y)
+     saida <- 2 * (n * log(mu/mu.hat) + (mu^-1 - mu.hat^-1) *
+         sum(y^2))
+     return(saida - desloca)
+ }
> dev.app.theta <- function(theta, theta.hat, y) {
+     n <- length(y)
+     saida <- (theta - theta.hat)^2 * (n/theta.hat^2)
+     return(saida)
+ }
> dev.app.mu.obs <- function(mu, mu.hat, y) {
+     n <- length(y)
+     Io <- -((n/mu.hat^2) - (2 * sum(y^2)/mu.hat^3))
+     saida <- (mu - mu.hat)^2 * Io
+     return(saida)
+ }
> dev.app.mu.esp <- function(mu, mu.hat, y) {
+     n <- length(y)
+     Ie <- n/(mu.hat^2)
+     saida <- (mu - mu.hat)^2 * Ie
+     return(saida)
+ }
```

Como é possível ver na Figura 1.7 a função *deviance* apresenta um comportamento bastante assimétrico o que torna a aproximação quadrática ruim, é claro que o tamanho da amostra reduzido também prejudica a aproximação quadrática.

De acordo com as propriedades do estimador de máxima verossimilhança, sabemos que $\hat{\theta} \sim N(\theta, I_E(\theta)^{-1})$, assintóticamente podemos usar ao invés de $I_E(\theta)$ tanto a $I_E(\hat{\theta})$ como também $I_O(\hat{\theta})$. Sabemos também que para θ a informação esperada e a observada coincidem. Então o intervalo assintótico fica dado por:

$$\hat{\theta}_L = \hat{\theta} - Z_{\frac{\alpha}{2}}\sqrt{\frac{\hat{\theta}^2}{n}} \quad \text{e} \quad \hat{\theta}_L = \hat{\theta} - Z_{\frac{\alpha}{2}}\sqrt{\frac{\hat{\theta}^2}{n}}$$

O mesmo argumento assintótico é usado para construir o intervalo para μ, porém aqui a informação esperada difere da observada o que neste caso não faz tanta diferença temos que a informação esperada é dada por $I_E(\hat{\mu}) = \frac{\mu^2}{n}$ enquanto que a informação observada é dada por $I_O(\hat{\mu}) = -\frac{n}{\mu^2} + \frac{2\sum_{i=1}^{N} x_i^2}{\mu^3}$, e o intervalo é construído exatamente igual ao anterior.

Para os intervalos baseados diretamente na função *Deviance* precisamos resolver as respectivas equações não lineares. Com os dados observados podemos fazer isso facilmente usando a função *uniroot.all*(). Vamos obter os intervalos de confiança e fazer uma comparação qualitativas em ambas as parametrizações. Para isto vamos criar uma função genérica que recebe a estimativa de máxima verossimilhança, a informação esperada ou observada e o nível de confiança do intervalo e retorno o seu limites inferior e superior.

```
> ic.assintotico <- function(EMV, Io, alpha) {
+     ic <- c(EMV - qnorm(1 - alpha/2) * sqrt(Io^(-1)), EMV + qnorm(1 -
```

```
+         alpha/2) * sqrt(Io^(-1)))
+     return(ic)
+ }
```

Usando a função criada vamos obter os intervalos assintóticos para θ e μ.

```
> n <- length(y)
> theta.est <- n/sum(y^2)
> ic.theta <- ic.assintotico(EMV = theta.est, Io = (n/theta.est^2),
+     alpha = 0.05)
> mu.est <- sum(y^2)/n
> ic.mu.obs <- ic.assintotico(EMV = mu.est, Io = -n/mu.est^2 +
+     (2 * sum(y^2))/(mu.est^3), alpha = 0.05)
> ic.mu.esp <- ic.assintotico(EMV = mu.est, Io = n/mu.est^2, alpha = 0.05)
```

Vamos também criar uma função genérica para à qual tem como seu argumento principal uma função com *Deviance* do modelo.

```
> ic.deviance <- function(dev, intervalo, ...) {
+     ic <- uniroot.all(dev, interval = intervalo, ...)
+     return(ic)
+ }
```

Usamos a função criada para obter os intervalos de confiança,

```
> ic.dev.theta <- ic.deviance(dev = dev.theta, intervalo = c(0,
+     10), theta.hat = theta.est, y = y, desloca = qchisq(0.95,
+     df = 1))
> ic.dev.mu <- ic.deviance(dev = dev.mu, intervalo = c(0, 100),
+     mu.hat = mu.est, y = y, desloca = qchisq(0.95, df = 1))
```

Vamos comparar os intervalos obtidos,

```
> rbind(ic.theta, ic.dev.theta)
```

```
                [,1]      [,2]
ic.theta     0.1521416 0.6481721
ic.dev.theta 0.2005100 0.7018490
```

```
> rbind(ic.dev.mu, ic.mu.obs, ic.mu.esp)
```

```
               [,1]     [,2]
ic.dev.mu 1.4248010 4.987302
ic.mu.obs 0.9501398 4.047900
ic.mu.esp 0.9501398 4.047900
```

É claro que em situações prática não é necessário percorrer todo o caminho feito neste exemplo, porém neste momento julgamos necessário o desenvolvimento completo do tópico, já que, reparametrização é muito comum quando ajusta-se modelos mais complexos. Considere que temos apenas as estimativas pontuais e intervalares de θ e desejamos obter as estimativas pontual e intervalar para μ. Temos que $\theta = \frac{1}{\mu}$, logo $\mu = \frac{1}{\theta}$ por invariância $\hat{\mu} = \frac{1}{\hat{\theta}}$. Os

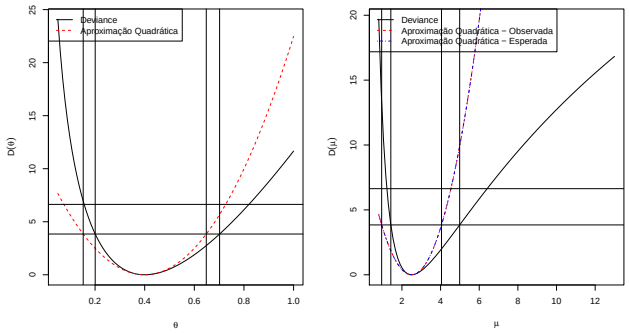

Figura 1.7: Diferentes formas de visualizar a função de verossimilhança.

intervalos obtidos diretamente baseado na função *Deviance* também são invariantes então basta aplica a transformação inversa igual a estimativa pontual. Quando usamos a aproximação quadrática os intervalos não são invariantes, é neste ponto que usamos o Teorema 4. No exemplo, obtemos o intervalo aproximado para θ usando por exemplo a informação esperada e desejamos o intervalo para μ. Usando o Teorema 4 sabemos que $\mu = g(\theta)$ e conhecemos a $V(\hat{\theta}) = \frac{\hat{\theta}^2}{n}$, partindo disto precisamos encontrar a variância para $\hat{\mu}$, o Teorema 4 diz que $V(\hat{\mu}) = g'(\hat{\theta})^2 I_E(\hat{\theta})^{-1}$, onde $g'()$ representa a primeira derivada da função $g()$. Sendo assim,

$$\hat{\mu} = \frac{1}{\hat{\theta}} = g(\hat{\theta}) \rightarrow g'(\hat{\theta}) = -\frac{1}{\hat{\theta}^2}$$

Logo, temos que

$$\begin{aligned} V(\hat{\mu}) &= \left(-\frac{1}{\hat{\theta}^2}\right)\frac{\hat{\theta}^2}{n} \\ &= \frac{1}{\hat{\theta}^4}\frac{\hat{\theta}^2}{n} = \frac{1}{\hat{\theta}^2 n} \\ &= = \frac{1}{0.400^2 * 10} = 0.625 \end{aligned}$$

Fazendo as contas no R,

```
> V.mu <- 1/(theta.est^2 * n)
> ic.mu.theta <- c(1/theta.est - qnorm(0.975) * sqrt(V.mu), 1/theta.est +
+     qnorm(0.975) * sqrt(V.mu))
```

Comparando com o intervalo obtido anteriormente,

```
> cbind(ic.mu.theta, ic.mu.esp)

     ic.mu.theta ic.mu.esp
[1,]   0.9501398 0.9501398
[2,]   4.0479002 4.0479002
```

Os intervalos são idênticos com um esforço muito menor. O Teorema 4 as vezes é chamado de método Delta para obter variância de estimadores. Estes resultados dos estimadores de máxima verossimilhança são muito utilizados quando estamos programando modelos mais complexos, principalmente modelos com efeitos aleatórios, onde diversos parâmetros de variância/precisão serão estimados. De forma geral, estes tipos de estimadores vão apresentar na parametrizaao original, uma distribuição amostral bastante assimétrica, além de problemas de representação numérica, tornando o uso de algoritmos numéricos dificil. Reparametrizações em termos de raízes e logaritmo são muito utilizadas para estabilizar os algoritmos numéricos e tornar a distribuição amostral dos estimadores mais próxima da gaussiana, o que também ajuda para a construção de intervalos baseados na aproximação quadrática. Sendo que a volta para a parametrização original pode ser feita pelo método Delta, para manter as interpretações desejadas em termos dos parâmetros originais do modelo. Vimos neste exemplo também, que mesmo em situações simples os intervalos baseados da função *Deviance* são de dificil obtenção necessitando de algoritmos numéricos para sua obtenção, porém de forma geral são mais realistas, no sentido de representar melhor a incerteza associada a estimação do parâmetro.

1.9 Exemplo 5 - Distribuição Normal

Suponha que $Y_1, Y_2, \ldots, Y_n$ são variáveis aleatórias independentes e identicamente distribuídas com distribuição Normal de média μ e variância σ^2. Denote isto por $Y_i \sim N(\mu, \sigma^2)$. Note que neste caso o vetor de parâmetros $\underline{\theta} = (\mu, \sigma)^\top$. Onde $\mu \in \Re$ e $\sigma \in \Re^+$ são os respectivos espaços paramétricos. O objetivo é estimar μ e σ além de encontrar intervalos ou regiões de confiança. Note que agora estamos trabalhando com uma superficie de log-verossimilhança o que tende a complicar a construção de gráficos e a obtenção das estimativas como um todo. Vejamos alguns fatos relevantes deste exemplo, como em tudo que fizemos neste livro, começamos escrevendo a função de verossimilhança,

$$\begin{aligned} L(\mu, \sigma) &= \prod_{i=1}^{n} \frac{1}{(2\pi)^{-1/2}\sigma} \exp^{-\frac{1}{2\sigma^2}(y_i-\mu)^2} \\ &= (2\pi)^{-n/2}\sigma^{-n} \exp^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\mu)^2}. \end{aligned}$$

A log-verossimilhança é dada por,

$$l(\mu, \sigma) = -\frac{n}{2}\log 2\pi - n\log\sigma - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mu)^2.$$

Note que agora a função *escore* toma a forma de um sistema de equações,

$$\begin{aligned} U(\mu) &= \frac{\partial l(\mu, \sigma)}{\partial \mu} = \frac{\sum_{i=1}^{n} y_i}{\sigma^2} - \frac{n\mu}{\sigma^2} \\ U(\sigma) &= -\frac{n}{\sigma} + \frac{1}{\sigma^3}\sum_{i=1}^{n}(y_i - \mu)^2 \end{aligned}$$

Podemos facilmente resolver este sistema chegando as estimativas de máxima verossimilhança,

$$\hat{\mu} = \frac{\sum_{i=1}^{n} y_i}{n} \quad \text{e} \quad \hat{\sigma} = \frac{\sum_{i=1}^{n}(y_i - \mu)^2}{n}.$$

A matriz de informação observada fica da seguinte forma,

$$\begin{bmatrix} -\frac{\partial^2 l(\mu,\sigma)}{\partial \mu^2} & -\frac{\partial^2 l(\mu,\sigma)}{\partial \mu \partial \sigma} \\ -\frac{\partial^2 l(\mu,\sigma)}{\partial \mu \partial \sigma} & -\frac{\partial^2 l(\mu,\sigma)}{\partial \sigma^2} \end{bmatrix}.$$

Temos então,

$$\begin{aligned} \frac{\partial^2 l(\mu,\sigma)}{\partial \mu^2} &= \frac{\partial U(\mu)}{\partial \mu} = -\frac{n}{\sigma^2} \\ \frac{\partial^2 l(\mu,\sigma)}{\partial \sigma^2} &= \frac{\partial U(\sigma)}{\partial \sigma} = -\frac{2n}{\sigma^2} \\ \frac{\partial^2 l(\mu,\sigma)}{\partial \mu \partial \sigma} &= \frac{\partial U(\sigma)}{\partial \sigma} = -\frac{2}{\sigma^3}\sum_{i=1}^{n}(y_i - \overline{y}) = 0. \end{aligned}$$

Logo,

$$I_O(\hat{\mu},\hat{\sigma}) = \begin{bmatrix} \frac{n}{\hat{\sigma}^2} & 0 \\ 0 & \frac{2n}{\hat{\sigma}^2} \end{bmatrix}.$$

Neste caso a matriz de informação observada coincide com a matriz de informação esperada. Além disso, note a importante propriedade de ortogonalidade entre os parâmetros, indicada pelos termos fora da diagonal da matriz de informação serem zero. Sempre que a derivada cruzada entre dois parâmetros for zero, dizemos que estes parâmetros são ortogonais. Grosseiramente falando a ortogonalidade é uma propriedade muito desejada, uma vez que podemos fazer inferência para um parâmetro sem nos preocupar com os valores do outro.

Para construção dos intervalos de confiança, a maneira mais direta é usar os resultados assintóticos dos estimadores de máxima verossimilhança, neste caso temos que a distribuição assintótica de $\underline{\hat{\theta}} = (\hat{\mu},\hat{\sigma})^\top$ é

$$\begin{bmatrix} \hat{\mu} \\ \hat{\sigma} \end{bmatrix} \sim NM_2\left(\begin{bmatrix} \mu \\ \sigma \end{bmatrix}, \begin{bmatrix} \hat{\sigma}^2/n & 0 \\ 0 & \hat{\sigma}^2/2n \end{bmatrix}\right)$$

Logo intervalos de confiança de Wald podem ser obtidos facilmente

$$\hat{\mu} \pm Z_{\alpha/2}\sqrt{\frac{\hat{\sigma}^2}{n}}$$

e para σ temos

$$\hat{\sigma} \pm Z_{\alpha/2}\sqrt{\frac{\hat{\sigma}^2}{2n}}$$

Outra opção é obter uma região de confiança baseada na *Deviance*,

$$\begin{aligned} D(\mu,\sigma) &= 2[l(\hat{\mu},\hat{\sigma}) - l(\mu,\sigma)] \\ &= 2[n\log\left(\frac{\sigma}{\hat{\sigma}}\right) + \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\mu)^2 - \frac{1}{2\hat{\sigma}^2}\sum_{i=1}^{n}(y_i-\hat{\mu}). \end{aligned}$$

A *Deviance* aproximada tem a seguinte forma

$$D(\mu, \sigma) \approx (\underline{\theta} - \hat{\underline{\theta}})^\top I_o(\hat{\underline{\theta}})(\underline{\theta} - \hat{\underline{\theta}})$$

Note que neste caso a superfície de log-verossimilhança em duas dimensões esta sendo aproximada por uma elipse. É bastante intuitivo pensar que aproximar uma função em duas dimensões é mais dificil que em uma, sabemos também que esta aproximação será tanto melhor quanto maior for o tamanho da amostra. Para exemplificar esta idéia a Figura **??** apresenta o gráfico da função *Deviance* bidimensional para o caso do modelo Normal, de acordo com quatro tamanhos de amostra.

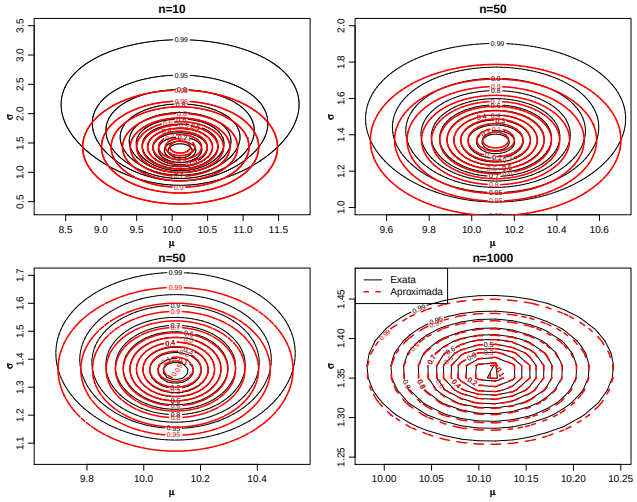

Figura 1.8: Diferentes formas de visualizar a função de verossimilhança.

Fica bastante claro pelos gráficos apresentados na Figura **??** que com um $n = 10$ a aproximação quadrática apresenta um comportamento bastante ruim, note a forte assimetria na direção do parâmetro σ. Conforme o tamanho da amostra vai aumentando a aproximação quadrática vai ficando cada vez mais próxima da *Deviance* exata, conforme esperado mais uma vez mostrando o comportamento assintótico do EMV. É importante notar também que a aproximação melhora mais rapidamente para μ do que para σ que precisa de um tamanho de amostra muito grande para que a *Deviance* em sua direção tome um comportamento próximo do simétrico. Isso mostra claramente que para parâmetros de média a aproximação quadrática tende a apresentar bons resultados mesmo com amostras reduzidas. O mesmo não pode ser dito para parâmetros de variância, isso mostra também que estimar média é mais simples que estimar variabilidade. As regiões de confiança são as curvas de nível na superfície de *Deviance*. Note que apenas com a *Deviance* não temos intervalos marginais como os obtidos pela aproximação quadrática. É claro que podemos olhar na superfície na direção do parâmetro de interesse na maior amplitude, que é obtida quando fixamos o outro parâmetro na sua estimativa de máxima verossimilhança. Porém como já foi dito anteriormente esta abordagem ignora a incerteza associada na estimação do parâmetro fixado. No caso da distribuição Normal, que tem a propriedade de ortogonalidade entre μ e σ, a verossimilhança condicionada na estimativa

de máxima verossimilhança coincide com a verossimilhança perfilhada. Para ilustrar este fato considere a obtenção da verossimilhança perfilhada para μ e σ pelas funções a seguir:

```
> perf.mu <- function(sigma, mu, dados) {
+     ll <- sum(dnorm(dados, mean = mu, sd = sigma, log = TRUE))
+     return(ll)
+ }
> perf.sigma <- function(mu, sigma, dados) {
+     ll <- sum(dnorm(dados, mean = mu, sd = sigma, log = TRUE))
+     return(ll)
+ }
```

Vamos criar um gride para avaliação da função e posterior construção dos gráficos. Também vamos obter a log-verossimilhança condicionada na estimativa de maxima verossimilhança, que consiste em apenas avaliar a função para um dos parâmetros com o outro fixado em sua estimativa.

```
> grid.mu <- seq(9, 11, l = 100)
> grid.sigma <- seq(1, 2, l = 100)
> vero.cond.mu <- sapply(grid.mu, perf.sigma, sigma = sd(y50),
+     dados = y50)
> vero.cond.sigma <- sapply(grid.sigma, perf.mu, mu = mean(y50),
+     dados = y50)
```

Para avaliar obter o perfil de verossimilhança, por exemplo para σ precisamos de uma grade de valores de σ e para cada valor nesta grade entrar o valor digamos $\hat{\mu}_\sigma$ que maximiza a verossimilhança perfilhada. Para evitar a obtenção de muitas equações vamos ilustrar o procedimento numericamente, para a maximização usamos a função *optimize*() própria para maximização em apenas uma dimensão como é o caso neste exemplo. O código abaixo ilustra o procedimento:

```
> vero.perf.mu <- c()
> for (i in 1:length(grid.mu)) {
+     vero.perf.mu[i] <- optimize(perf.mu, c(0, 200), mu = grid.mu[i],
+         dados = y50, maximum = TRUE)$objective
+ }
> vero.perf.sigma <- c()
> for (i in 1:length(grid.sigma)) {
+     vero.perf.sigma[i] <- optimize(perf.sigma, c(0, 1000), sigma = grid.sigma[i],
+         dados = y50, maximum = TRUE)$objective
+ }
```

Para finalizar basta desenhar os gráficos sobrepostos,

A figura 1.9 ilustra claramente a *Deviance* perfilhada e a *Deviance* condicional que neste caso devido a propriedade de ortogonalidade do modelo gaussiano coincidem. Para obter intervalos de confiança basta dar o corte nestas funções na altura do quantil da distribuição Qui-Quadrado com o respectivo nível de confiança e encontrar as raízes da equação, o que pode ser feito como nos exemplos unidimensionais. A verossimilhança perfilhada permite tratar um problema multidimensional como um problema unidimensional levando em consideração a incerteza associada na estimação de todos os parâmetros envolvidos

no modelo. Porém, como pode ser visto esta abordagem pode ser extremamente cara computacionalmente, uma vez que para cada avaliação da verossimilhança perfilhada pode ser necessário uma maximização, que em geral vai usar algum método numérico.

1.10 Exemplo 6 - Distribuição Gama

Sejam $Y_1, Y_2, \ldots, Y_n$ variáveis aleatórios independentes com distribuição Gama de parâmetros a e s. Nosso objetivo é partindo de uma amostra aleatório $y_1, y_2, \ldots y_n$ estimar os parâmetros a e s com seus respectivos intervalos de confiança baseados na aproximação quadrática e na função *Deviance*. A função de densidade da Gama tem a seguinte forma:

$$f(y) = \frac{1}{s^a \Gamma(a)} y^{a-1} \exp^{-y/s}, \quad \text{para} \quad y \geq 0 \quad \text{e} \quad a, s \geq 0.$$

Nesta parametrização $E(Y) = a * s$ e $V(Y) = a * s^2$. A função de verossimilhança é

$$\begin{aligned} L(a,s) &= \prod_{i=1}^{n} (s^a \Gamma(a))^{-1} y_i^{a-1} \exp^{-y_i/s} \\ &= s^{na} \Gamma^{-n} \exp^{1/s \sum_{i=1}^{n} y_i} \prod_{i=1}^{n} \log y_i^{a-1}. \end{aligned}$$

A função de log-verossimilhança

$$l(a,s) = -na \log s - n \log \Gamma(a) - \frac{1}{s} \sum_{i=1}^{n} y_i + (a-1) \sum_{i=1}^{n} \log y_i.$$

As funções *escore* sã obtidas derivando a log-verossimilhança em função dos respectivos parâmetros e são dadas por

$$\begin{aligned} U(a) &= -n \log s - n \frac{\Gamma'(a)}{\Gamma(a)} + \sum_{i=1}^{n} \log y_i \\ U(s) &= -\frac{na}{s} + \frac{1}{s^2} \sum_{i=1}^{n} y_i. \end{aligned}$$

Para obter as estimativas de máxima verossimilhança precisamos resolver o sistema de equações acima, em relação aos parâmetros a e s. Este sistema claramente não tem solução analitica em a, porém para s podemos facilmente encontrar que $\hat{s} = \frac{\overline{y}}{a}$. Substituindo $\hat{s}$ na função de log-verossimilhança, obtemos o que chamamos de log-verossimilhança concentrada em a. Note que em um primeiro momento tinhamos um problema de maximização em duas dimensões, com o uso da log-verossimilhança concentrada diminuimos o problema de maximação para apenas uma dimensão o que com certeza é mais eficiente computacionalmente. A expressão da log-verossimilhança concentrada é dada por:

$$l_a(s) = -na \log \frac{\overline{y}}{a} - n \log \Gamma(a) - \frac{a}{\overline{y}} \sum_{i=1}^{n} y_i + a \sum_{i=1}^{n} \log y_i - \sum_{i=1}^{n} \log y_i$$

que é uma função apenas do parâmetro a. Para encontrar a estimativa de a ainda precisamos maximizar a log-verossimilhança concentrada. Podemos para isto usar um otimizador numérico como o implementado na função *optimizer*() ou alguns dos métodos da função *optim*(). Mas antes disso, vamos investigar como é o relacionamento entre a e s em termos de ortogonalidade. Para isto, precisamos obter a matriz, neste caso 2×2 de derivadas segundas, ou seja, a matriz de informação observado e/ou esperada.

Derivando as funções escore temos,

$$\begin{aligned}
\frac{\partial^2 l(a,s)}{\partial a^2} &= -n \left[\frac{\Gamma(a)\Gamma''(a) - \Gamma'(a)^2}{\Gamma(a)^2} \right] \\
\frac{\partial^2 l(a,s)}{\partial s^2} &= \frac{na}{s^2} - \frac{2}{s^3} \sum_{i=1}^{n} y_i \\
\frac{\partial^2 l(a,s)}{\partial a \partial s} &= -\frac{n}{s}.
\end{aligned}$$

Logo a matriz de informação observada é dada por,

$$I_o(a,s) = \begin{bmatrix} n \left[\frac{\Gamma''(a)}{\Gamma'(a)} - \left(\frac{\Gamma'(a)}{\Gamma(a)} \right)^2 \right] & \frac{n}{s} \\ \frac{n}{s} & -na/s^2 + 2/s^3 \sum_{i=1}^{n} y_i \end{bmatrix}$$

A matriz esperada é obtida tomando a esperança da matriz observada, lembrando que $E(Y) = a * s$ temos

$$I_E(a,s) = \begin{bmatrix} n \left[\frac{\Gamma''(a)}{\Gamma(a)} - \left(\frac{\Gamma'(a)}{\Gamma(a)} \right)^2 \right] & \frac{n}{s} \\ \frac{n}{s} & \frac{na}{s^2}. \end{bmatrix}$$

Como mostram as matrizes de informação os termos fora da diagonal são não-nulos o que mostra que os parâmetros são não ortogonais. Para visualizarmos o formato da função de log-verossimilhança a Figura **??** apresenta a superfície de log-verossimilhança e sua aproximação quadrátic em escala de *Deviance* para facilitar a construção e visualização do gráfico.

Pelos gráficos podemos ver claramente que quando o tamanho da amostra é pequeno $n = 10$ o formato da log-verossimilhança é extremamente assimétrico e consequentemente a aproximação quadrática é muito ruim. Com o aumento da amostra a aproximação quadrática vai melhorando, até que com $n = 2000$ a diferença é bastante pequena. Os gráficos também mostram a dependência entre os parâmetros a e s, quando o a aumenta necessariamente o s diminui para manter a média que é $a * s$, além disso fica claro também que a incerteza associada a estimativa de a é muito maior quando comparada a estimativa de s.

Agora que temos todos os ingredientes podemos nos concentrar em obter as estimativas de máxima verossimilhança. Lembre-se que a log-verossimilhança concentrada é dada por

$$l_a(s) = -na \log \frac{\overline{y}}{a} - n \log \Gamma(a) - \frac{a}{\overline{y}} \sum_{i=1}^{n} y_i + a \sum_{i=1}^{n} \log y_i - \sum_{i=1}^{n} \log y_i$$

que é uma função apenas do parâmetro a. Uma vez obtido a estimativa $\hat{a}$ podemos substitui-la em $\hat{s} = \frac{\overline{y}}{\hat{a}}$ e obter a estimativa de s. Da mesma forma podemos substituir as estimativas nas matrizes de informação observada e esperada e encontrar intervalos de confiança assintóticos, sabendo que estes intervalos serão consistentes apenas com amostras grandes. Mas para todos estes procedimentos precisamos maximizar a log-verossimilhança concentrada em a. A forma mais comum de fazer isso é usando o algoritmo de Newton-Raphson ou uma variante deste chamada de algoritmo Escore de Fisher. Vamos abrir um parenteses e explicar rapidamente o algoritmo de Newton-Raphson.

O método de Newton-Raphson é usado para se obter a solução numérica de uma equação na forma $f(x) = 0$, onde $f(x)$ é contínua e diferenciável e sua equação possui uma solução próxima a um ponto dado. O processo de solução começa com a escolha do ponto x_1 como a primeira tentativa de solução. A segunda tentativa, x_2, é obtida a partir do cruzamento com o eixo x da reta tangente a $f(x)$ no ponto $(x_1, f(x_1))$. A tentativa seguinte, x_3 é a intersecção com o eixo x da reta tangente a $f(x)$ no ponto $(x_2, f(x_2))$, e assim por diante. A equação de iteração é dada por:

$$x_{i+1} = x_i - \frac{f(x_i}{f'(x_i)} \tag{1.2}$$

ela é chamada de equação de iteração porque a solução é obtida com a aplicação repetida em cada valor sucessivo de i até que um critério de convergência seja atingido. Diversos critérios de convergência podem ser usados. Os mais comuns são:

- Erro relativo
$$\left| \frac{x_{i+1} - x_i}{x_i} \right| \leq \epsilon$$
- Tolerância em $f(x)$
$$|f(x_i) \leq \delta$$

A programação do método de Newton é muito simples. Uma função chamada

$$NewtonRaphson()$$

é apresentada abaixo.

```
> NewtonRaphson <- function(initial, escore, hessiano, tol = 1e-04,
+     max.iter, n.dim, ...) {
+     solucao <- matrix(NA, max.iter, n.dim)
+     solucao[1, ] <- initial
+     for (i in 2:max.iter) {
+         solucao[i, ] <- initial - solve(hessiano(initial, ...),
+             escore(initial, ...))
+         initial <- solucao[i, ]
```

```
+         tolera <- abs(solucao[i, ] - solucao[i - 1, ])
+         print(initial)
+         if (all(tolera < tol) == TRUE)
+             break
+     }
+     return(initial)
+ }
```

Note que para usar este algoritmo é necessário obter a primeira (escore) e a segunda (hessiano) derivada. O que neste exemplo é possível, porém em modelos mais complexos pode não ser, ou mesmo, o custo computacional em calcular o hessiano analitico pode ser muito maior que o numérico, o que acontece em alguns modelos multivariados que em geral envolvem muitas inversões de matrizes densa, fazendo com que este algoritmo se torne muito lento.

Cabe ressaltar que o método de Newton-Raphson é um algoritmo para encontrar raízes de uma equação que no caso da função escore leva as estimativas de máxima verossimilhança. Porém existem diversos e poderosos algoritmos de maximização numerica que não precisam de derivadas analiticas, porém podem ser muito beneficiados com o uso de resultados destas principalmente a função escore. No R alguns destes maximizadores numéricos estão implementados na função *optim*().

Continuando com o exemplo da Gama, vamos obter a função escore e o hessiano da função de log-verossimilhança concentrada e usar o algoritmo de Newton-Raphson para obter a estimativa de a. Derivando a log-verossimilhança em a obtemos,

$$U_c(a) = \sum_{i=1}^{n} \log y_i - n \log \frac{\overline{y}}{a} - n\psi_0(a) + \sum_{i=1}^{n} \log y_i$$

onde $\psi_0(a)$ é a função digama que é a derivada primeira do logaritmo da função gama. Derivando novamente em a obtemos,

$$U_c'(a) = \frac{n}{a} - n\psi_1(a)$$

onde $\psi_1(a)$ é a função trigama que é a derivada segunda do logaritmo da função gama.

Escrevendo estas funções em R

```
> escore <- function(a, y) {
+     n <- length(y)
+     u <- -n * log(mean(y)/a) - n * digamma(a) + sum(log(y))
+     return(u)
+ }
> hessiano <- function(a, y) {
+     n <- length(y)
+     u.l <- (n/a) - trigamma(a) * n
+     return(u.l)
+ }
```

O passo final para obter a estimativa de máxima verossimilhança de a é usar o algoritmo de Newton Raphson.

```
> a.hat <- NewtonRaphson(initial = 5, escore = escore, hessiano = hessiano,
+     max.iter = 100, n.dim = 1, y = y10)

[1] 8.09194
[1] 11.32100
[1] 13.16977
[1] 13.52671
[1] 13.53677
[1] 13.53678

> a.hat

[1] 13.53678
```

O algoritmo conforme programado mostra todas as tentativas do algoritmo até a convergência. Neste exemplo foi necessário seis iterações para atingir o critério de convergência. Uma limitação deste algoritmo é que o chute inicial não pode estar muito longe da solução, o que pode ser dificil de obter em modelos complexos, nestes casos estimativas grosseiras por minimos quadrados podem ser de grande valia como chute inicial.

Uma vez estimado o a podemos substituir na expressão de $\hat{s}$ para obtê-lo,

```
> s.hat <- mean(y10)/a.hat
> s.hat

[1] 3.614117
```

Para construção dos intervalos assintóticos basta substituir as estimativas nas matrizes de informação observada e/ou esperada. Note que no caso da distribuição Gama a distribuição assintótica do vetor $(\hat{a}, \hat{s})^\top$ é a seguinte,

$$\begin{bmatrix}\hat{a}\\ \hat{s}\end{bmatrix} \sim NM_2\left(\begin{bmatrix}a\\ s\end{bmatrix}, \begin{bmatrix}n\psi_1(\hat{a}) & n/\hat{s}\\ n/\hat{s} & n\hat{a}/\hat{s}^2\end{bmatrix}\right)^{-1}$$

poderiamos usar também a matriz de informação observada que é assintóticamente equivalente. O código abaixo implementa da matriz de informação esperada e observada e contrói os intervalos de confiança assintóticos para a e s.

```
> Ie <- function(a, s, y) {
+     n <- length(y)
+     saida <- matrix(c(n * trigamma(a), n/s, n/s, (n * a)/s^2),
+         2, 2)
+     return(saida)
+ }
> Io <- function(a, s, y) {
+     n <- length(y)
+     saida <- matrix(c(n * trigamma(a), n/s, n/s, -(n * a)/s^2 +
+         (2/s^3) * sum(y)), 2, 2)
+     return(saida)
+ }
```

Avaliando as matrizes no ponto de máximo,

```
> Ie(a = a.hat, s = s.hat, y = y10)

          [,1]      [,2]
[1,]  7.666854  27.66927
[2,] 27.669273 103.63604

> Io(a = a.hat, s = s.hat, y = y10)

          [,1]      [,2]
[1,]  7.666854  27.66927
[2,] 27.669273 103.63604
```

Como é possível observar a matriz de informação esperada e observada apesar de apresentarem formas diferentes levam a resultados idênticos para o tamanho de amostra $n = 100$ considerado aqui. Sendo assim, vamos usar apenas a informação esperada que é mais simples de avaliar. Para obter os intervalos assintóticos basta inverter a matriz de informação e pegar os termos em sua diagonal que corresponde a variância de $\hat{a}$ e $\hat{s}$.

```
> erro.padrao <- sqrt(diag(solve(Ie(a = a.hat, s = s.hat, y = y10))))
> ic.a <- a.hat + c(-1, 1) * qnorm(0.975) * erro.padrao[1]
> ic.s <- s.hat + c(-1, 1) * qnorm(0.975) * erro.padrao[2]
> ic.a

[1]  9.829956 17.243600

> ic.s

[1] 2.605898 4.622336
```

Outra forma é a construção de intervalos baseados no perfil de verossimilhança. As funções apresentadas abaixo implementam o perfil de verossimilhança para os parâmetros a e s respectivamente.

```
> perf.a <- function(s, a, dados) {
+     ll <- sum(dgamma(dados, shape = a, scale = s, log = TRUE))
+     return(ll)
+ }
> perf.s <- function(a, s, dados) {
+     ll <- sum(dgamma(dados, shape = a, scale = s, log = TRUE))
+     return(ll)
+ }
```

Para as maximizações necessárias vamos utilizar a função *optimize*() própria para maximização em uma dimensão como é o caso aqui. Precisamos também criar uma grade para a avaliação da função. Também será avaliada a verossimilhança condicional para comparação conforme realizado no Exemplo 5.

```
> grid.a <- seq(9, 18, l = 100)
> grid.s <- seq(2, 5, l = 100)
> vero.perf.a <- c()
> for (i in 1:length(grid.a)) {
```

```
+     vero.perf.a[i] <- optimize(perf.a, c(0, 200), a = grid.a[i],
+         dados = y10, maximum = TRUE)$objective
+ }
> vero.perf.s <- c()
> for (i in 1:length(grid.s)) {
+     vero.perf.s[i] <- optimize(perf.s, c(0, 1000), s = grid.s[i],
+         dados = y10, maximum = TRUE)$objective
+ }
> vero.cond.a <- sapply(grid.a, perf.a, s = s.hat, dados = y10)
> vero.cond.s <- sapply(grid.s, perf.s, a = a.hat, dados = y10)
```

Para ilustrar os resultados a Figura **??** apresenta os resultados.

Como mostra a Figura 1.11 os intervalos obtidos condicionando a log-verossimilhança na estimativa de máxima verossimilhança são claramente mais curtos que o intervalo baseado em perfil de verossimilhança e o intervalo assintótico. Comparando o perfil com o assintótico verificamos que a perfilhada traz intervalos ligeiramente assimétricos e mais largos que a aproximação quadrática, a aproximação é ligeiramente deslocada para esquerda e para direita para os parâmetros a e s respectivamente.

Neste exemplo passamos detalhadamente por cada passo do processo de inferência. Apresentamos o algoritmo de Newton Raphson e comparamos três abordagens para a construção de intervalos de confiança. Neste exemplo, ficou claro o uso intensivo de ferramentas do cálculo diferencial para obter a função escore e a matriz de informação. Ficou claro também que nem sempre é fácil obter estas quantidades analiticamente.

1.11 Exemplo 7 - Distribuição Binomial Negativa

Considere $Y_1, Y_2, \ldots, Y_n$ variáveis aleatórios independentes provenientes de uma distribuição Binomial Negativa de parâmetros $\phi \in \Re^+$ e $0 < p \leq 1$ e $y = 0, 1, \ldots$. Como em tudo até aqui estamos interessados através de uma amostra $y_1, y_2, \ldots, y_n$ estimar os parâmetros ϕ e p e possivelmente construir intervalos de confiança e testes de hipóteses. Neste caso temos dois parâmetros, o que leva a uma superfície de verossimilhança. A função de distribuição da Binomial Negativa é dada por:

$$p(y) = \frac{\gamma(y+\phi)}{\gamma(\phi)y!}p^{\phi}(1-p)^{y}, \quad \text{para} \quad 0 < p \leq 1 \quad \phi > 0 \quad \text{e} \quad y = 0, 1, 2, \ldots$$

Sendo n amostras independentes a função de verossimilhança tem a seguinte forma,

$$\begin{aligned} L(\phi, p) &= \prod_{i=1}^{n} \frac{\gamma(y_i+\phi)}{\gamma(\phi)y_i!}p^{\phi}(1-p)_i^{y} \\ L(\phi, p) &= p^{n\phi}(1-p)^{\sum_{i=1}^{n} y_i} \prod_{i=1}^{n} \frac{\gamma(y_i+\phi)}{\gamma(\phi)y_i!}. \end{aligned}$$

Logo a função de log-verossimilhança pode ser escrita como,

$$l(\phi, p) = n\phi \log p + \sum_{i=1}^{n} y_i \log(1-p) + \sum_{i=1}^{n} \log \gamma(y_i + \phi) - n \log \gamma(\phi) - \sum_{i=1}^{n} \log y_i!.$$

Derivando em relação aos parâmetros ϕ e p chegamos as equações escore.

$$\begin{aligned} U(\phi) &= n \log p + \sum_{i=1}^{n} \psi_0(y_i + \phi) - n\psi_0(\phi) \\ U(p) &= \frac{n\phi}{p} - \sum_{i=1}^{n} y_i/(1-p) \end{aligned}$$

Para obter a matriz de informação precisamos das derivadas segundas que são:

$$\begin{aligned} \frac{\partial^2 l(\phi,p)}{\partial \phi^2} &= \frac{\partial U(\phi)}{\partial \phi} = \frac{\partial}{\partial \phi} n \log p \sum_{i=1}^{n} \psi_0(y_i+\phi) - n\psi_0(\phi) = \sum_{i=1}^{n} \psi_1(y_i+\phi) - n\psi_1(\phi) \\ \frac{\partial^2 l(\phi,p)}{\partial p^2} &= \frac{\partial U(p)}{\partial p} = \frac{\partial}{\partial p} \frac{n\phi}{p} - \sum_{i=1}^{n} y_i/(1-p) \\ \frac{\partial l(\phi,p)}{\partial \phi \partial p} &= \frac{\partial U(\phi)}{\partial p} = \frac{\partial}{\partial p} n \log p + sum_{i=1}^{n} \psi_0(y_i+\phi) - n\psi_0(\phi). \end{aligned}$$

Dado estes resultados temos todos os ingredientes para estimar ϕ e p através do algoritmo de Newton Raphson. Note que pela equação $U(p)$ podemos obter a estimativa de p em função do parâmetro desconhecido ϕ da mesma forma que no modelo Gama. Porém, neste exemplo não vamos usar verossimilhança concentrada para exemplificar o uso do algoritmo de Newton Raphson em duas dimensões.

A implementação do modelo começa sempre com a construção da função de log-verossimilhança, neste caso tal qual escrevemos no papel. Adiante veremos como usar as funções residentes do *R* para implementar isso de forma mais eficiente. Vamos também simular uma amostra de tamanho $n = 100$ desta distribuição para imitar uma realização do modelo, vamos fixar os valores dos parâmetros como $\phi = 100$ e $p = 0.8$.

```
> set.seed(123)
> y <- rnbinom(100, size = 100, p = 0.8)
> ll.neg.binomial <- function(par, y) {
+     phi <- par[1]
+     p <- par[2]
+     n <- length(y)
+     ll <- n * phi * log(p) + sum(y) * log(1 - p) + sum(log(gamma(y +
+         phi))) - n * log(gamma(phi)) - sum(log(factorial(y)))
+     return(ll)
+ }
```

Pensando em otimizar a função de log-verossimilhança usando o algoritmo de Newton Raphson o próximo passo é implementar a função escore.

```
> escore <- function(par, y) {
+     phi <- par[1]
+     p <- par[2]
+     n <- length(y)
+     U.phi <- n * log(p) + sum(digamma(y + phi)) - n * digamma(phi)
+     U.p <- (n * phi)/p - sum(y)/(1 - p)
+     return(c(U.phi, U.p))
+ }
```

Na sequência a matriz de segundas derivadas ou Hessiana.

```
> Hessiana <- function(par, y) {
+     phi = par[1]
+     p = par[2]
+     n <- length(y)
+     II <- matrix(c(sum(trigamma(y + phi)) - n * trigamma(phi),
+         n/p, n/p, -(n * phi)/p^2 - sum(y)/(1 - p)^2), 2, 2)
+     return(II)
+ }

> NewtonRaphson(initial = c(50, 0.5), escore = escore, hessiano = Hessiana,
+     max.iter = 100, tol = 1e-10, n.dim = 2, y = y)

[1] 42.9758078  0.6211639
[1] 40.2896150  0.6186758
[1] 54.1105521  0.6995206
[1] 56.3186131  0.6939266
[1] 73.1908653  0.7575637
[1] 75.457782  0.752376
[1] 90.4447124  0.7893347
[1] 92.005714  0.787418
[1] 105.180532   0.811381
[1] 106.1645382   0.8103905
[1] 112.7042384   0.8198534
[1] 113.0335133   0.8198349
[1] 114.166931   0.821316
[1] 114.1935819   0.8213381
[1] 114.1976243   0.8213433
[1] 114.1976249   0.8213433
[1] 114.1976249   0.8213433
[1] 114.1976249   0.8213433
[1] 114.1976249   0.8213433
[1] 114.1976249   0.8213433
```

Para construção do intervalo de confiança assintótico e/ou baseado em perfil de verossimilhança podemos proceder exatamente igual ao exmeplo da distribuição Gama. Deixamos como exercício para o leitor obter estes intervalos e compará-los.

1.12 Tratando tudo numericamente

Vimos nos exemplos anteriores que métodos numéricos são essenciais em inferência baseado em verossimilhança. Mesmo em exemplos simples com apenas dois parâmetros soluções analiticas não podem ser obtidas. Vimos também que o algoritmo de Newton Raphson é muito poderoso para resolver sistemas do tipo $f(x) = 0$, porém ele necessita do vetor escore e da matriz de derivadas segundas (Hessiana). O que nem sempre pode ser fácil de ser obtida e mesmo em exemplos simples gera um trabalho analitico consideravel.

Além disso, vimos que a implementação de intervalos baseados em perfil de verossimilhança é um tanto quanto tediosa, mesmo os intervalos de Wald que são simples exigem de um tempo razoavel de programação que precisa ser muito cuidadosa. Nesta seção nos vamos abordar os mesmo três problemas a dois parâmetros já apresentados porém tratando eles de forma inteiramente numérica. Para isto, vamos usar a função *optim*() que implementa quatro poderosos algoritmos de otimização numérica, são eles Nelder Mead, Gradiente Conjugado, Simulating Annealing e BFGS.

Porém, só esta função não resolve o problema da construção dos intervalos de confiança e testes de hipóteses. Para isto, vamos introduzir o pacote *bbmle* que traz uma verdadeira caixa de ferramentas para inferência baseada em verossimilhança. Nos vamos ver que com pouca programação conseguiremos estimar os parâmetros, construir intervalos de confiança assintóticos e perfilhados, obter testes de hipóteses, além de obter os resultados de uma forma elegante, como a saída padrão de funções como a *lm*() e *glm*().

Veremos também como os resultados analiticos já obtidos, principalmente o vetor escore podem ser usados para acelerar e melhorar a performance dos otimizadores numéricos. Relembre que no exemplo 5, tratamos o modelo Normal com média μ e variância σ^2. Podemos escrever a o **negativo** da log-verossimilhança deste modelo em R da seguinte forma:

```
> ll.gauss <- function(par, dados) {
+     ll <- sum(dnorm(dados, par[1], sd = par[2], log = TRUE))
+     return(-ll)
+ }
```

Note que escrevemos o negativo da log-verossimilhança, isso é meramente computacional porque por padrão a função *optim*() minimiza a função objetivo. Observe que passamos os parâmetros a ser estimados como um vetor unico em que cada posição representa um parâmetro a ser estimado. Para proceder o processo de estimação precisamos de uma amostra, vamos simular uma amostra de tamanho $n = 100$ do modelo Normal com $\mu = 0$ e $\sigma = 1$ apenas para ilustrar o procedimento via a função *optim*(). Na maioria dos algoritmos de otimização numérica será necessário o uso de chutes iniciais, para isto é comum utilizar alguma estimativa grosseira ou mesmo fazer várias tentativas e avaliar a sensibilidade do algoritmo as condições iniciais.

Um cuidado que se deve ter quando se usa este tipo de algoritmos numéricos e o espaço de busca, que no caso de inferência estatística corresponde ao espaço paramétrico. Para o parâmetro μ a busca deve ser em toda a reta real, porém para σ apenas nos reais positivos. A maioria dos algoritmos da *optim*() não leva isso em consideração, fazendo com que tente avaliar a função de verossimilhança

em pontos não válidos. Este problema pode gerar desde simples mensagens de *warnings* até a falha total do algoritmo. Reparametrizações são muito utéis neste ponto. O único algoritmo dentro da *optim*() que permite busca em espaços determinados é o $L-BFGS-B$ é o que vamos usar neste exemplo. Recomendamos que o leitor leia a documentação da função e experimente os outros algoritmos.

```
> set.seed(123)
> dados <- rnorm(100, 0, 1)
> est.gauss <- optim(par = c(0.5, 10), fn = ll.gauss, dados = dados,
+     method = "L-BFGS-B", lower = c(-Inf, 1e-32), upper = c(Inf,
+         Inf), hessian = TRUE)
> est.gauss

$par
[1] 0.09040592 0.90824121

$value
[1] 132.2692

$counts
function gradient
      19       19

$convergence
[1] 0

$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"

$hessian
              [,1]          [,2]
[1,]  1.212265e+02 -4.114042e-06
[2,] -4.114042e-06  2.424550e+02
```

A saída da *optim*() retorna uma lista. O primeiro elemento é o valor que maximiza o negativo da log-verossimilhança em seguida o valor que ela toma neste ponto, muito importante para a construção de testes de hipóteses e comparações de modelos, na sequência diversas informações osbre o procedimento e por fim a matriz Hessiana obtida numericamente, importante para a construção dos intervalos de confiança assintóticos. Deste ponto, ainda é preciso uma quantidade razoável de programação para obtenção de intervalos perfilhados. Uma forma mais rápido de obter praticamente tudo que se deseja do processo de inferência é usar o pacote *bbmle*, que facilita todo o procedimento.

Para usar este pacote precisamos escrever a função de log-verossimilhança ligeiramente diferente. O código abaixo apresenta as funções de log-verossimilhança para os casos Gaussianos, Gama e Binomial Negativo.

```
> ll.gauss <- function(mu, sigma, dados) {
+     ll <- sum(dnorm(dados, mu, sigma, log = TRUE))
+     return(-ll)
```

```
+ }
> ll.gamma <- function(a, s, dados) {
+     ll <- sum(dgamma(dados, shape = a, scale = s, log = TRUE))
+     return(-ll)
+ }
> ll.negbin <- function(phi, p, dados) {
+     ll <- sum(dnbinom(dados, size = phi, p = p, log = TRUE))
+     return(-ll)
+ }
```

A estimação via o pacote *bbmle* é feita através da função *mle*2(), esta função é apenas uma casca para a *optim*() veja um exemplo de chamada para o modelo Gaussiano.

```
> est.gauss <- mle2(ll.gauss, start = list(mu = 0.5, sigma = 100),
+     data = list(dados = dados), method = "L-BFGS-B", lower = list(mu = -Inf,
+         sigma = 1e-32), upper = list(mu = Inf, sigma = Inf))
```

As opções básicas vão desde um simples resumo do modelo ajustado via função *summary*(),

```
> summary(est.gauss)

Maximum likelihood estimation

Call:
mle2(minuslogl = ll.gauss, start = list(mu = 0.5, sigma = 100),
    method = "L-BFGS-B", data = list(dados = dados), lower = list(mu = -Inf,
        sigma = 1e-32), upper = list(mu = Inf, sigma = Inf))

Coefficients:
      Estimate Std. Error z value  Pr(z)
mu    0.090407   0.090828  0.9954 0.3196
sigma 0.908243   0.064223 14.1421 <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

-2 log L: 264.5385
```

Construção de intervalos de confiança assintóticos.

```
> confint(est.gauss, method = "quad")

           2.5 %   97.5 %
mu    -0.0876118 0.268426
sigma  0.7823690 1.034118
```

Construção de intervalos de confiança perfilhados.

```
> confint(est.gauss)
```

```
Profiling...
            2.5 %    97.5 %
mu    -0.08934198 0.2701538
sigma  0.79559101 1.0503091
```

O código baixo mostra o procedimento para o caso Gama, note a definição dos intervalos de busca para o algoritmo $L--BFGS-B$.

```
> set.seed(123)
> dados.gama <- rgamma(100, shape = 10, scale = 1)
> est.gamma <- mle2(ll.gamma, start = list(a = 2, s = 10), data = list(dados = dados.gama),
+     method = "L-BFGS-B", lower = list(a = 1e-32, s = 1e-32),
+     upper = list(a = Inf, s = Inf))
> summary(est.gamma)

Maximum likelihood estimation

Call:
mle2(minuslogl = ll.gamma, start = list(a = 2, s = 10), method = "L-BFGS-B",
    data = list(dados = dados.gama), lower = list(a = 1e-32,
        s = 1e-32), upper = list(a = Inf, s = Inf))

Coefficients:
  Estimate Std. Error z value     Pr(z)
a 13.53632    1.89105  7.1581 8.181e-13 ***
s  0.72285    0.10288  7.0263 2.120e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

-2 log L: 474.394

> confint(est.gamma)

Profiling...
      2.5 %     97.5 %
a 10.164953 17.5943060
s  0.553859  0.9687491

> confint(est.gamma, method = "quad")

      2.5 %     97.5 %
a 9.8299293 17.2427185
s 0.5212131  0.9244835
```

O mesmo para o modelo Binomial Negativo.

```
> set.seed(123)
> dados.nbin <- rnbinom(1000, size = 10, p = 0.8)
> est.nbin <- mle2(ll.negbin, start = list(phi = 2, p = 0.5), data = list(dados = dados.nbin),
+     method = "L-BFGS-B", lower = list(phi = 1e-32, p = 1e-32),
+     upper = list(phi = Inf, p = 0.99999))
> summary(est.nbin)
```

```
Maximum likelihood estimation

Call:
mle2(minuslogl = ll.negbin, start = list(phi = 2, p = 0.5), method = "L-BFGS-B",
    data = list(dados = dados.nbin), lower = list(phi = 1e-32,
        p = 1e-32), upper = list(phi = Inf, p = 0.99999))

Coefficients:
    Estimate Std. Error z value     Pr(z)
phi 8.634386   1.735262  4.9758 6.497e-07 ***
p   0.772486   0.035543 21.7339 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

-2 log L: 3853.87
```

```
> confint(est.nbin)

Profiling...
        2.5 %     97.5 %
phi 6.0825513 13.7049387
p   0.7043744  0.8436871
```

```
> confint(est.nbin, method = "quad")

       2.5 %    97.5 %
phi 5.233335 12.035437
p   0.702823  0.842149
```

O esforço de programação quando tratamos tudo numéricamente é menor. O uso das funções do pacote *bbmle* facilita muito o trabalho de inferência, pois já traz pronto tudo que é genérico do método além de formatar a saída da forma padrão do R. É claro que isto torna a inferência computacionalmente mais cara.

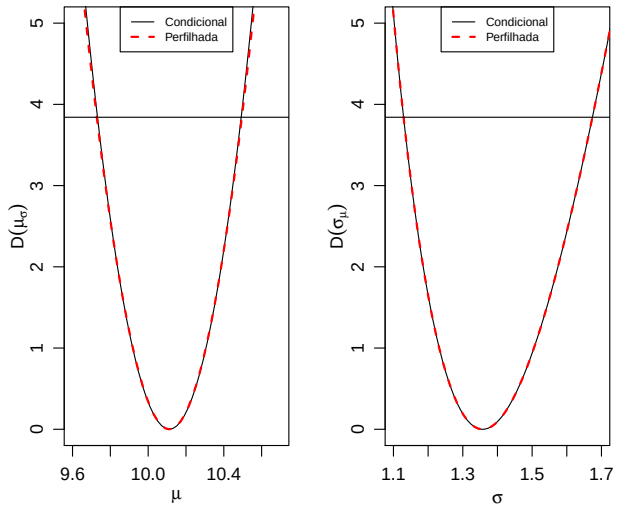

Figura 1.9: Deviance perfilhada e condicional para μ e σ - Modelo Gaussiano.

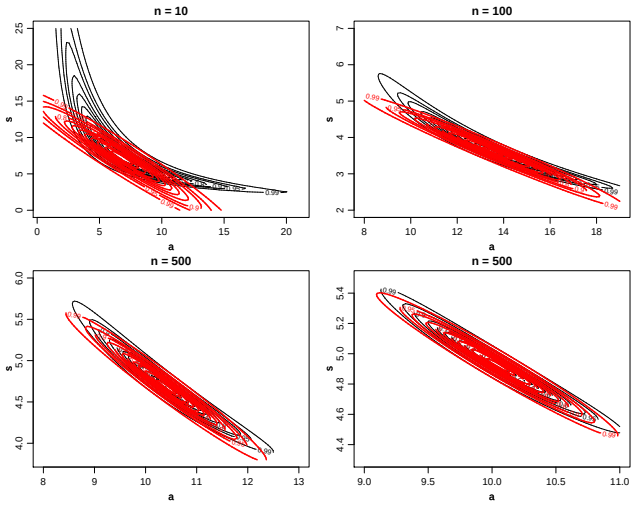

Figura 1.10: Diferentes formas de visualizar a função de verossimilhança.

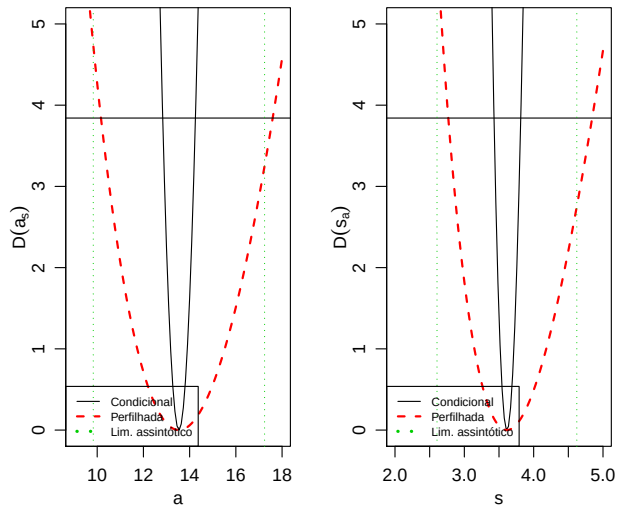

Figura 1.11: Deviance perfilhada, condicional e limites do intervalo de confiança assintótico para a e s - Modelo Gama.

Capítulo 2

Modelos de regressão

Modelos estocásticos têm sido amplamente utilizados tanto na comunidade científica como no mundo dos negócios em geral. Estudos de mercado usando modelos altamente estruturados, modelos de predição em séries financeiras, análises de componentes de solos para agricultura de precisão, mapeamento de doenças são algumas das principais área de aplicações de tais modelos.

A área de modelos de regressão teve seu início com o tradicional modelo de regressão linear gaussiano, que foi por décadas utilizado para todas as situações onde se desejava estudar o relacionamento entre um conjunto de covariáveis e uma variável resposta. Diversas transformações na resposta foram propostas para adaptar a suposição de normalidade, porém são modelos não satisfatórios para resposta discreta como no caso de dados binários, contagens baixas e respostas restritas a certos intervalos como proporções e indices restritos ao intervalo unitário.

A modelagem estatística recebeu um grande impulso desde a criação dos modelos lineares generalizados no início da década de 70 [**?**]. Variáveis resposta binárias e de contagens foram as que receberam mais atenção, talvez pela dificuldade em tratar deste tipo de resposta com tranformações do modelo linear gaussiano e hoje os modelos de regressão logística e de Poisson, são muito utilizados nas mais diversas áreas de pesquisa.

Considere que $Y_1, Y_2, \ldots, Y_n$ são variáveis aleatórias independentes e identicamente distribuídas e X é a matriz de delineamento do modelo conhecida. Um modelo de regressão em notação matricial pode ser escrito da seguinte forma:

- $Y|X \sim f(\underset{\sim}{\mu}, \phi)$;
- $\underset{\sim}{\mu} = g(X; \underset{\sim}{\beta})$.

Nesta forma de apresentação a distribuição de probabilidade $f(\underset{\sim}{\mu}, \phi)$ é descrita por dois conjuntos de parâmetros, os de locação (média) e os de dispersão (precisão / variância). Note que neste caso estamos supondo que o parâmetro de dispersão é comum a todas as observações o que muda é apenas a média. Esta restrição pode ser facilmente retirada, supondo que também o parâmetro de dispersão, seja alguma função de covariáveis igual a parte de média, optamos por esta restrição por simplicidade, porém cabe ressaltar que é possível e de fácil implementação computacional.

Nesta distribuição é muito importante estar atento ao espaço paramétrico de $\underline{\mu}$ e ϕ. O segundo em geral tem como seu espaço paramétrico os reais positivos, já que, é um parâmetro de variabilidade tem que ser necessariamente positivo. Para a parte de média devemos ter mais cuidado, uma vez que a função $g(.)$ deve mapear o preditor linear ou não-linear que pode assumir qualquer valor real, para o espaço paramétrico de $\underline{\mu}$. Por exemplo, se estamos trabalhando com dados de contagens onde o vetor $\underline{\mu}$ representa o parâmetro da distribuição de Poisson, a função $g(.)$ deve mapear dos reais para os reais positivos. Em outra situação, por exemplo com dados restritos ao intervalo $(0, 1)$ a função $g(.)$ deve mapear dos reais para o intervalo unitário e assim por diante.

Note que para a declaração completa do modelo é necessário duas suposições, a primeira se refere a distribuição de probabilidade atribuída a variável resposta, neste caso $f(\underline{\mu}, \phi)$ e de uma função $g(.)$ que faz com que o preditor linear ou não-linear, que pode apresentar qualquer valor nos reais, seja mapeado para o espaço paramétrico da parte de média do modelo.

A função $g(.)$ é a princípio apenas duplamente diferenciável, uma restrição adicional comum é que seja estritamente monotona, porém em modelos não-lineares isso nem sempre é necessário. Esta função tem como seus argumentos a matriz de delineamento X conhecida e os parâmetros $\underline{\beta}$ desconhecidos a serem estimados. Sendo a função $g(.)$ desta forma e mapeando os valores do preditor seja ele linear ou não para espaço paramétrico de $\underline{\mu}$ o modelo está composto e é passível de ter seus parâmetros estimados pela máximização da função de verossimilhança.

Um fato que pode passar desapercebido é com relação ao suporte da distribuição concordar ou não com os possíveis valores do fenômeno em estudo. Por exemplo, ao estudar o peso animais é comum atribuir o modelo Gaussiano, porém este modelo tem como suporte os reais, enquanto que o fenômeno varia apenas nos reais positivos. Nestes casos a usual justificativa prática é que a probabilidade atribuída a parte negativa é tão pequena que é desprezível na prática.

Uma outra situação é quando trabalhamos com uma variável resposta restrita ao intervalo unitário, podemos definir como função $g(.)$ o inverso da função *logit* e vamos estar mapeando a média do modelo ao intervalo unitário o que concorda com os dados. Porém, qunado escolhemos a distribuição nem sempre isso é feito, é comum por exemplo em modelos não-lineares atribuir para $f(\underline{\mu}, \phi)$ a dsitribuição Gaussiano que tem suporte nos reais o que não concorda com os dados. Apesar deste tipo de construção ser muito comum, é mais natural atribuir uma distribuição de probabilidade que tenha suporte concordante com o campo de variação da variável resposta. No caso de resposta restrita ao intervalo unitário, a distribuição Beta ou Simplex seriam opções razoáveis. Quando pensamos na concordância entre dados e modelos é mais natural atribuir ao fenômeno uma distribuição cujo suporte, concorde com o campo de variação do fenômeno.

Dado a relevância de modelos de regressão em aplicações nas mais diversas áreas do conhecimento, neste capítulo vamos mostrar como implementar a estimação de alguns modelos de regressão tradicionais, como regressão de Poisson, Simplex e não-linear Gaussiano. Vamos explorar o algoritmo de Newton-Raphson no modelo de regressão Poisson, para o modelo Simplex, vamos explorar o conceito de verossimilhança concentrada e exemplificar como usar a função *escore* obtida analiticamente dentro do algoritmo BFGS implementado

na função *optim*(). No último modelo vamos tratar tudo numéricamente apenas como uma implementação simplificada. Neste capítulo como material extra apresentamos a implementação computacional de modelos dinâmicos e um estudo de caso de modelos de regressão para contagens subdispersas.

2.1 Regressão Poisson

Em situações onde a variável resposta é discreta na forma de contagens, um modelo de regressão muito utilizado é o de Poisson. Conforme vimos anteriormente um modelo de regressão quaçquer é descrito por duas suposições: a distribucional que se refere a distribuição de probabilidade atribuída a variável resposta e a uma função $g(.)$ que liga o preditor linear ou não a média desta distribuição.

No caso do modelo de Poisson, como o próprio nome já diz para a primeira suposição usamos a distribuição de Poisson que tem a seguinte forma,

$$P(Y = y) = \frac{\exp^{-\lambda} \lambda^y}{y!}, \quad \text{onde} \quad \lambda \geq 0 \quad \text{e} y = 0, 1, 2 \ldots.$$

Conforme a construção do modelo Poisson seu parâmetro indexador λ corresponde a esperança de Y, neste modelo temos apenas a componente de média, já que a variância é igual a média. O parâmetro λ deve ser maior que zero, logo devemos escolher uma função $g(.)$ que mapeie dos reais aos reais positivos. Uma função muito usada em regressão Poisson é a exponencial. No contexto de modelos lineares generalizados a função de ligação canônica do modelo Poisson é a função logaritmo cuja a inversa é a exponencial, por isso su popularidade. Resumindo, supomos que $\underline{Y} \sim P(\underline{\lambda})$ e que $\underline{\lambda} = \exp^{X\underline{\beta}}$ com a suposição adicional que o preditor é linear. Com isso, chegamos ao modelo de regressão de Poisson.

A função de log-verossimilhança escrita em formato matricial é dada por,

$$l(\underline{\beta}, \underline{y}) = -1^\top g(X\underline{\beta}) + \underline{y}^\top X\underline{\beta} + 1^\top \log(\underline{y}!).$$

Derivando em relação ao vetor de parâmetros $\underline{\beta}$ temos a função escore.

$$U(\underline{\beta}) = \{\underline{y} - \exp^{x\underline{\beta}}\}^\top X$$

Derivando novamente obtemos a matriz de informação observada.

$$I_o(\underline{\beta}) = X^\top [-diag(X\underline{\beta})]X$$

Com isso temos todos os elementos para compor o algoritmo de Newton Raphson e encontrar as estimativas de máxima verossimilhança para o vetor $\underline{\beta}$. Para exemplificar o processo vamos tratar com dados simulados e vamos supor que $X\underline{\beta}$ tem a forma $\beta_0 + \beta_1 x_i$ onde x_i é uma covariável conhecida. Assim o modelo tem apenas dois parâmetros $\underline{\beta} = (\beta_0, \beta_1)$. O código abaixo apresenta uma função para simular realizações deste modelo.

```
> simula.poisson <- function(b0, b1, n.simul) {
+     cov <- seq(0, 5, l = n.simul)
+     X <- model.matrix(~cov)
+     eta <- X %*% c(b0, b1)
+     lambda <- exp(eta)
```

```
+     y <- rpois(n.simul, lambda = lambda)
+     return(data.frame(y = y, cov = cov))
+ }
```

A Figura 2.1 apresenta a superfície de log-verossimilhança em escala de *Deviance* exata e pela aproximação quadrática, para diferentes tamanhos de amostras.

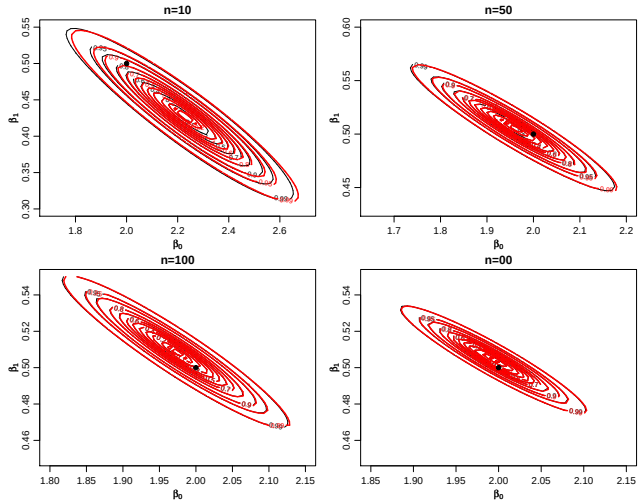

Figura 2.1: Superficie de deviance para diferentes tamanhos de amostra - Regressão Poisson.

A Figura mostra claramente que a aproximação quadrática é excelente para os dois parâmetros, o que era esperado, já que, se tratam de parâmetros de média. A dependencia entre os dois parâmetros também é clara nos gráficos mostrando que os parâmetros não são ortogonais. Note que com o aumento do tamanho da amostra os contornos vão ficando menores, mais concentrados em torno do verdadeiro valor do parâmetro, evidenciando a propriedade assintótica de não-viés.

Seguindo com o processo de inferência podemos usar o algoritmo de Newton-Raphson para encontrar as estimativas de máxima verossimilhança de β_0 e β_1, para isto precisamos escrever duas funções, a *escore* e a *hessiana* isso é feito no código abaixo.

```
> escore <- function(par, formula, dados) {
+     X <- model.matrix(as.formula(formula), data = dados)
+     esco <- t(dados$y - exp(X %*% c(par[1], par[2]))) %*% X
+     return(esco)
+ }
> hessiano <- function(par, formula, dados) {
+     X <- model.matrix(as.formula(formula), data = dados)
+     mat <- diag(length(dados$y))
+     diag(mat) <- -exp(X %*% c(par[1], par[2]))
+     H <- t(X) %*% mat %*% X
```

```
+     return(H)
+ }
```

Usando o algoritmo de Newton-Raphson já apresentado, temos

```
> estimativa <- NewtonRaphson(initial = c(0, 0), escore = escore,
+     hessiano = hessiano, max.iter = 100, n.dim = 2, formula = "~cov",
+     dados = dados200)
> estimativa

[1] 1.9945610 0.5042655
```

Para a construção dos intervalos de confiança assintóticos, basta avaliar a matriz Hessiana no ponto encontrado,

```
> Io <- -hessiano(par = estimativa, formula = "~cov", dados = dados200)
> Io

            (Intercept)      cov
(Intercept)     6688.00 23157.29
cov            23157.29 90674.85
```

Lembrando que as variâncias de $\hat{\beta}_0$ e $\hat{\beta}_1$ são dadas pelos termos da diagonal da inversa da matriz de informação observada, temos pela distribuição assintótica do EMV que um intervalo de 95% de confiança para β_0 e β_1 é dado por

```
> desvio.padrao <- sqrt(diag(solve(Io)))
> estimativa[1] + c(-1, 1) * qnorm(0.975) * desvio.padrao[1]

[1] 1.924107 2.065015

> estimativa[2] + c(-1, 1) * qnorm(0.975) * desvio.padrao[2]

[1] 0.4851313 0.5233996
```

Com isso, exemplificamos a construção do modelo de regressão de Poisson, procedemos a estimação por máxima verossimilhança, obtivemos o vetor *escore* e a matriz *hessiana* que possibilitaram o uso do algoritmo de Newton-Raphson. Vimos também que a aproximação quadrática no caso do modelo Poisson é bastante acurada, sendo que visualmente é dificil encontrar diferença entre ela e a exata. Usando resultados assintóticos construímos intervalos de confiança. Além disso, com os resultados obtidos estamos aptos a proceder testes de hipóteses, por exemplo, $H_0 : \beta_1 = 0$ contra $H_1 : \beta_1 \neq 0$. Onde podemos usar o tradicional teste de Wald, ou mesmo o teste de razão de verossimilhança.

2.2 Regressão Simplex

Em diversas situações práticas estamos interessados na análise de variáveis restritas ao intervalo $(0, 1)$, como taxas, proporções e índices. Nas ciências sociais é comum a mensuração de variáveis latentes (não observáveis) que muitas vezes

buscam medir qualidade, através de indicadores indiretos sobre o fenômeno latente. Um exemplo deste tipo de análise é o IDH - Índice de Desenvolvimento Humano, preconizado pela ONU - Organização das Nações Unidas, em todo o mundo.

Estes índices que tradicionalmente por construção resultam sempre em valores no intervalo unitário, buscam medir indiretamente o nível de desenvolvimento de um país, ou adaptações deste como o IDH-M (Índice de Desenvolvimento Humano - Municipal) de municípios, estados ou qualquer aglomeração de indivíduos, como por exemplo, bairros de uma cidade. O importante em ressaltar que a construção leva a um número no intervalo unitário, como uma nota que serve para comparação entre países, estados, municípios ou bairros.

Com foco em tais situações, selecionamos para este exemplo o Índice de Qualidade de Vida de Curitiba (IQVC). Este índice é composto por 19 indicadores separados em 4 áreas temáticas: Habitação, Saúde, Educação e Segurança. Estes indicadores buscam de forma indireta medir o nível de vida da população residente em cada um dos 75 bairros da cidade. A metodologia para sua construção segue as premissas do IDH, e como tal resulta sempre em valores no intervalo unitário. Para sua construção é utilizada a base de microdados do Censo 2000, disponibilizada pelo IBGE (Instituto Brasileiro de Geografia e Estatística). Um interesse comum é relacionar o IQVC com a renda média do bairro medida em salários mínimos vigentes na época da pesquisa, neste caso ano de 2000. Os dados são disponibilidados em www.ippuc.org.br e uma versão resumida no arquivo *simplex.txt*.

Para a construção do modelo de acordo com o *framework* apresentado, precisamos fazer duas suposições, a primeira a respeito da distribuição de probabilidade atribuída a variável resposta e a segunda referente a função $g(.)$ que vai mapear o preditor linear ou não ao espaço paramétrico induzido pela distribuição suposta. Para este exemplo vamos supor a distribuição como sendo Simplex e a função $g(.)$ como a inversa da função $logit()$ comum em modelos lineares generalizados. A apresentação do modelo aqui segue o trabalho de [**?**].

A função densidade de probabilidade Simplex é indexada por dois tipos de parâmetros locação $\mu \in (0,1)$ e dispersão $\sigma^2 > 0$. Ambos podem ser descompostos em função de covariáveis observadas. Uma variável aleatória y que segue o modelo Simplex tem função densidade dada por

$$f(y;\mu,\sigma^2) = [2\pi\sigma^2 y(1-y)^3]^{-1/2} \exp\left[-\frac{1}{2\sigma^2} d(y;\mu)\right], \quad y \in (0,1), \qquad (2.1)$$

em que

$$d(y;\mu) = \frac{(y-\mu)^2}{y(1-y)\mu^2(1-\mu)^2}.$$

Sejam $Y_1, Y_2, \ldots, Y_n$ variáveis aleatórias indenpendentes, sendo cada $Y_i \sim S(\mu_i, \sigma^2)$, $i = 1, 2, \ldots, n$. O modelo de regressão Simplex é definido pela densidade 2.1, sendo as médias μ_i dadas por

$$\mu_i = g(x_i^\top \underline{\beta}) = g(\eta_i)$$

em que $g(.)$ é uma função que transforma valores dos reais ao intervalo unitário, $\underline{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)^\top$ é o vetor de parâmetros da regressão ($\underline{\beta} \in \Re^p$),

$x_i^\top = (x_{i1}, x_{i2}, \ldots, x_{ip})^\top$ são os valores conhecidos de p covariáveis e η_i neste caso é o preditor linear. Poderíamos, sem perda de generalidade definir o preditor como

$$h(\mu_i) = x_i^\top \underline{\beta} = \eta$$

onde agora $h(.)$ é uma função de ligação que transforma do $(0,1)$ em $\Re$, dependendo da situação pode ser mais simples definir de uma forma ou de outra. No caso Simplex vamos definir a função $h(.)$ como sendo a função *logit* que facilita bastante a obtenção do vetor *escore*. O objetivo deste exemplo é mostrar como usar os otimizadores numéricos implementados na função *optim*(), porém usando o vetor *escore* obtido analiticamente.

Dada a função densidade de probabilidade, podemos facilmente obter a função de log-verossimilhança,

$$l_i(\mu_i, \sigma^2) = -\frac{1}{2}\log(2\pi) - \frac{1}{2}\log(\sigma^2) - \frac{3}{2}\log\{y_i(1-y_i)\} - \frac{1}{2\sigma^2}d(y_i; \mu_i.$$

Temos que $h(\mu_i) = x_i^\top \underline{\beta} = \eta_i$, então para obter o vetor escore precisamos derivar em relação a cada $\underline{\beta}_p$.

$$\frac{\partial l(\underline{\beta}, \sigma^2)}{\partial \beta_p} = \sum_{i=1}^{n} \frac{\partial l_i(\mu_i, \sigma^2)}{\partial \mu_i}\frac{\partial \mu_i}{\partial \eta_i}\frac{\partial \eta_i}{\partial \beta_i}$$

Para obter o vetor escore para a parte de média precisamos de três componentes, o primeiro é

$$\begin{aligned}
\frac{\partial l(\mu_i, \sigma^2)}{\partial \mu_i} &= -\frac{1}{2\sigma^2}\frac{\partial}{\partial \mu_i}d(y_i, \mu_i) \\
&= \sigma^{-2}\left[\underbrace{\frac{(y_i-\mu_i)}{(1-\mu)^2\mu^2(1-y)y} + \frac{(y_i-\mu_i)^2}{(1-\mu)^2\mu^3(1-y)y} - \frac{(y_i-\mu_i)^2}{(1-\mu_i)^3\mu^2(1-y_i)y_i}}_{u}\right]
\end{aligned}$$

O segundo,

$$\frac{\partial \mu_i}{\partial \eta_i} = \frac{1}{g'(\mu_i)} = \mu_i(1-\mu_i)$$

E o terceiro,

$$\frac{\partial \eta_i}{\partial \beta_i} = x_{ip}.$$

Em notação matricial, a função escore para $\underline{\beta}$ é dada por

$$U_{\underline{\beta}}(\underline{\beta}, \sigma^2) = \sigma^{-2}X^\top T u,$$

onde X é uma matriz $n \times p$, de delineamento do modelo, $T = diag\{1/g'(\mu_1), \ldots, 1/g'(\mu_i)\}$ e $u^\top = (u_1, \ldots, u_n)\top$.

A função escore para σ^2 é facilmente obtida, derivando $l(\underline{\beta}, \sigma^2)$ em relação a σ^2,

$$U_{\sigma^2}(\underline{\beta}, sigma^2) = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{i=1}^{n} d(y_i; \mu_i).$$

Note que podemos obter $\hat{\sigma}^2$ analiticamente, sua equação é dada por

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n} d(y_i, \hat{\mu}_i)$$

Com isso temos todos os elementos nencessários para a implementação computacional do modelo de regressão Simplex. No R não tem uma densidade Simplex, então vamos começar a implementação por ela.

```
> dsimplex <- function(y, mu, sigma, log = TRUE) {
+     dis <- (y - mu)^2/(y * (1 - y) * mu^2 * (1 - mu)^2)
+     dsim <- -0.5 * log(2 * pi) - 0.5 * log(sigma) - (3/2) * log(y *
+         (1 - y)) - (1/(2 * sigma)) * dis
+     if (log == FALSE) {
+         dsim <- exp(dsim)
+     }
+     return(dsim)
+ }
```

Definimos a função $h(.) = g^{-1}(.)$ como a logit então a função $g(.)$ é implementada abaixo.

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

Dada a aplicação teremos que o vetor $\underline{\beta} = (\beta_0, \beta_1)$ então o modelo de regressão Simplex em R é apresentado no código abaixo. Note que como o parâmetro σ^2 tem solução analitica e depende apenas de μ_i não precisamos maximizar numéricamente em relação a este parâmetro, estamos novamente usando uma log-verossimilhança concentrada. Basta substituirmos seu estimador na expressão da log-verossimilhança.

```
> modelo.simplex <- function(b0, b1, formula, data) {
+     X <- model.matrix(as.formula(formula), data = data)
+     mu <- inv.logit(X %*% c(b0, b1))
+     d0 = (data$y - mu)^2/(data$y * (1 - data$y) * mu^2 * (1 -
+         mu)^2)
+     sigma <- sum(d0)/length(data$y)
+     ll <- sum(dsimplex(data$y, mu = mu, sigma = sigma))
+     return(-ll)
+ }
```

O próximo passo é implementar a função escore já obtida, note novamente que ela é função apenas de dois parâmetros β_0 e β_1, uma que tenhamos estes substituimos na expressão de $\hat{\sigma}^2$ e o usamos na expressão do vetor escore.

```
> escore <- function(b0, b1, formula, data) {
+     X <- model.matrix(as.formula(formula), data = data)
+     eta <- X %*% c(b0, b1)
+     mu <- inv.logit(eta)
+     d0 = (data$y - mu)^2/(data$y * (1 - data$y) * mu^2 * (1 -
+         mu)^2)
+     sigma <- sum(d0)/length(data$y)
+     T <- diag(c(mu * (1 - mu)))
+     u <- (1/(sigma * (1 - mu) * mu)) * (-((data$y - mu)^2/((1 -
+         mu)^2 * mu^2)) + ((data$y - mu)^2/((1 - mu)^2 * mu^2)) +
+         ((data$y - mu)/((1 - mu)^2 * mu^2)))
+     es <- sigma^-1 * t(X) %*% T %*% u
+     saida <- as.numeric(-es)
+     return(saida)
+ }
```

Com isto, estamos aptos a passar a log-verossimilhança e o vetor escore para a optim e obter as estimativas de máxima verossimilhança para β_0 e β_1, com estas substituímos na expressão de $\hat{sigma}^2$ e concluímos o processo de estimação. Antes disso, precisamos do conjunto de dados que será analisado. O arquivo chamado *simplex.txt* traz o conjunto para a análise. O código abaixo lê a base de dados e apresenta um pequeno resumo. A Figura 2.3 apresenta uma análise exploratória gráfica, com um histograma e um diagrama de dispersão relacionando o IQVC com a renda média do bairro em salários mínimos que foi divido por 10 para evitar problemas numéricos.

```
> dados <- read.table("simplex.txt", header = TRUE)
> head(dados)

  ID      y MEDIA
1 10 0.6416 1.471
2  6 0.7644 1.529
3 70 0.8580 1.906
4 69 0.6695 2.161
5 62 0.8455 1.730
6 61 0.7861 1.724
```

O conjunto de dados contêm apenas três colunas, a primeira denominada ID apenas identifica os diferentes bairros, a coluna y apresenta o Índice de Qualidade de Vida de Curitiba (IQVC) e a coluna $MEDIA$ apresenta a renda média do bairro em salários minímos dividido por 10.

Para o ajuste do modelo Simplex, vamos novamente usar as facilidades do pacote *bbmle* que usa internamente a função *optim*(). O ajuste é dado no seguinte código.

```
> require(bbmle)
> simplex.logit.gr <- mle2(modelo.simplex, start = list(b0 = 0,
+     b1 = 0), gr = escore, method = "BFGS", data = list(formula = "~MEDIA",
+     data = dados))
```

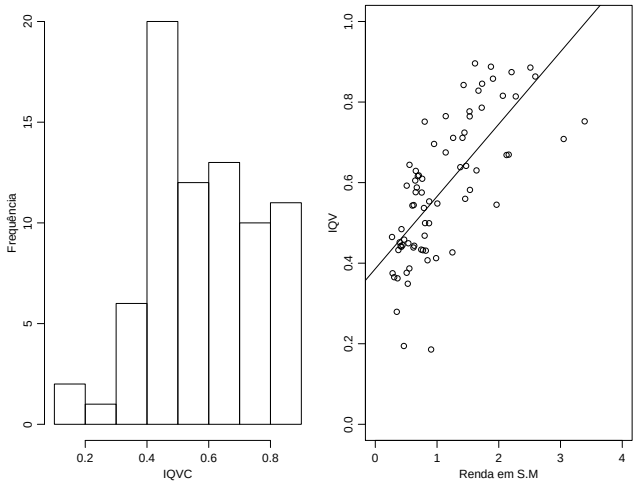

Figura 2.2: Histograma e diagrama de dispersão - IQVC.

A única diferença é no uso do argumento *gr* onde é passada a função escore com exatamente os mesmos argumentos adicionais usados na função de log-verossimilhança. A função *optim*() tem quatro otimizadores básicos: Nelder-Mead, Gradiente Conjugado, BFGS e Simulating Annealing. O uso de gradiente analitico só é possível nos algoritmos BFGS e Gradiente Conjugado.

O resumo tradicional do modelo pela função *summary*(), traz os erros padrões que são usados para a construção de intervalos de confiança assintóticos, que podem ser obtidos diretamente pela função $confint(..., method =' quad')$.

```
> summary(simplex.logit.gr)

Maximum likelihood estimation

Call:
mle2(minuslogl = modelo.simplex, start = list(b0 = 0, b1 = 0),
    method = "BFGS", data = list(formula = "~MEDIA", data = dados),
    gr = escore)

Coefficients:
    Estimate Std. Error z value     Pr(z)
b0 -0.399817   0.113700 -3.5164 0.0004374 ***
b1  0.683628   0.071231  9.5973 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

-2 log L: -103.1299

> confint(simplex.logit.gr, method = "quad")

        2.5 %     97.5 %
b0 -0.6226650 -0.1769692
b1  0.5440179  0.8232386
```

Deixamos como exercício para o leitor obter o intervalo de confiança para σ^2 e os perfis de verossimilhança para β_0 e β_1. Para finalizar o exemplo, podemos visualizar o ajuste sobreposto aos dados. Para isto, vamos escrever uma função genérica para gerar os preditos.

```
> my.predict <- function(modelo, formula, newdata) {
+     X <- model.matrix(as.formula(formula), data = newdata)
+     coefi <- coef(modelo)
+     beta <- coefi[1:dim(X)[2]]
+     preditos <- inv.logit(X %*% beta)
+     return(preditos)
+ }
```

Usando a função *my.predict*() obtemos os valores preditos pelo modelo para rendas em um gride de valores.

```
> preditos <- my.predict(simplex.logit.gr, formula = "~MEDIA",
+     newdata = data.frame(MEDIA = seq(0, 4, l = 500)))
```

Por fim, plotamos um diagrama de dispersão com as observações e a média predita pelo modelo de acordo com o gride de renda.

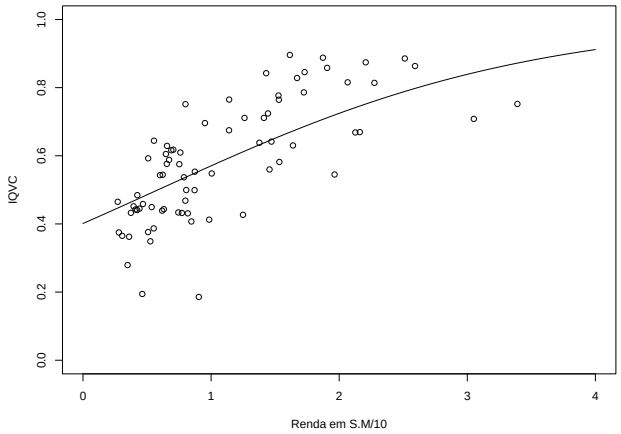

Figura 2.3: Diagrama de dispersão e modelo ajustado - IQVC 2000.

Capítulo 3

Modelos de regressão com efeitos aleatórios

A área de modelagem estatística teve um grande impulso com a criação dos modelos de regressão, conforme os apresentados no capítulo 2. As aplicações aparecem nas mais diversas áreas da ciência. Nesta diversidade de aplicações é muito fácil encontrar situações de relevância prática, onde os modelos de regressão tradicionais, deixam de ser adequados, em geral pela existência de pelo menos uma das seguintes características:

- para covariáveis contínuas, a suposição de efeito estritamente linear no preditor pode não ser adequada.
- as observações podem ser correlacionadas no espaço,
- as observações podem ser correlacionadas no temp,
- interações complexas podem ser necessárias para modelar o efeito conjunto de algumas covariáveis,
- heterogeneidade entre indivíduos ou unidades podem não ser suficientemente descrita por covariáveis.

Nestas situações a classe de modelos de regressão com efeitos aleatórios têm sido extensivamente usada, por ser altamente flexível. Esta classe de modelos pode facilmente ser descrita como uma extensão dos modelos de regressão com efeitos fixos, pela inclusão de mais uma suposição. Considere que Y seja um vetor de dimensão n. A seguinte estrutura hierárquica descreve um modelo de regressão com efeitos aleatórios. Neste livro nos vamos nos limitar a inclusão de efeitos aleatórios Gaussianos, como mostrado abaixo:

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

Note que nesta descrição o preditor linear é decomposto em duas componentes, a parte de efeitos fixos $X\underline{\beta}$ e a parte aleatória $Z\underline{b}$. As matrizes de

delineamente X e Z são conhecidas representando o efeitos de covariáveis de interesse. O vetor $\underline{\beta}$ representa os efeitos fixos que deverão ser estimados. O vetor aleatório $\underline{b}$ são quantidades não observadas (latentes) para o qual vamos atribuir uma distribuição Normal Multivariada de média 0 e matriz de covariância Σ. De acordo com a estrutura imposta a Σ podemos induzir diversos tipos de correlação entre as observações Y. É importante destacar a suposição de independência condicional, $Y|b$, ou seja, dado os efeitos aleatórios as observações são independentes o que permite facilmente construir a verossimilhança de $Y|b$.

A inferência baseada em verossimilhança para esta classe de modelos apresenta grandes desafios computacionais, principalmente quando a distribuição atribuída a variável resposta é diferente da Gaussiana. Para a seguinte discussão vamos sem perda de generalida exlcuir a matriz de efeitos fixos X. Como temos duas quantidades aleatórias nesta forma de construção, devemos obter a distribuição conjunta de $[Y, b]$ que pela estrutura hierárquica é simplesmente um produto,$[Y, b] = [Y|b][b]$ agora para fazer inferência temos apenas as observações de Y, então precisamos apenas da distribuição marginal de Y que avaliada nas observações realizadas torna-se a função de verossimilhança do modelo. Então o que precisamos obter é $[Y] = \int [Y|b][b]db$ e maximiza-la em relação as parâmetros do modelo. Conforme a suposição de que $[b]$ é Normal Multivariada, temos que os parâmetros do modelos são $[\underline{\beta}, \Sigma]$, ou seja o vetor de efeitos fixos mais os parâmetros que indexam a distribuição do efeito aleatório Gaussiano Multivariado.

O problema que surge é que a integral contida na função de verossimilhança quando o distribuição de $[Y]$ não é Gaussiana, na maioria das situações não tem solução analitica. Isto implica que métodos de integração numérica são necessários para avaliar a verossimilhança marginal e obter as estimativas de máxima verossimilhança. Esta descrição é bastante genérica e será detalhada na sequência. Mas antes vamos apresentar um exemplo onde a distribuição de $[Y]$ é Gaussiana porém as amostras não são independentes.

3.1 Modelo geoestatístico

O termo geoestatística, refere-se a modelos e métodos para dados seguindo as seguintes características: Os valores $Y_i : i = 1, \ldots, n$ são observados em um conjunto discreto de localizações amostrais,x_i , em alguma região espacial A. Cada valor é uma versão ruidosa de um fenômeno espacial contínuo não observável, denotado por $S(x)$, nas correspondentes localizações amostrais. O objetivo mais comum neste tipo de análise é recuperar o processo $S(x)$. Para isto, a abordagem padrão é o modelo geoestatístico, como apresentado em Diggle e Ribeiro Jr (2007). No framework apresentado, podemos identificar o modelo geoestatístico da seguinte forma:

- $Y|b, X \sim N(\underline{\mu}, I\tau^2)$
- $\underline{\mu} = X\underline{\beta} + Z\underline{b}$
- $\underline{b} \sim NMV(\underline{0}, \Sigma_b)$

onde X é uma matriz de covariáveis conhecidas e $\underline{\beta}$ o vetor de parâmetros associados a elas. Z é um vetor de uns. A parte deste modelo que merece mais atenção

é a matriz Σ_b, pois ela descreve a estrutura da dependência espacial entre as observações. Diversas opções para construir esta matriz são possíveis, neste exemplo vamos usar a função de correlação exponencial. A matriz considerando apenas duas localizações espaciais, toma a seguinte forma:

$$\Sigma_b = \begin{bmatrix} \sigma^2 & \sigma^2 \exp(-\frac{u_{12}}{\phi}) \\ \sigma^2 \exp(-\frac{u_{21}}{\phi}) & \sigma^2 \end{bmatrix}$$

onde u_{12} é a distância euclidiana entre as amostras $Y(x_1)$ e $Y(x_2)$, a matriz completa depende da matriz de distância entre todas as amostras espaciais e os parâmetros σ^2 e ϕ. O parâmetro ϕ controla o grau da dependência espacial entre as observações, enquanto que o parâmetro σ^2 é a variância deste efeito. Além disso, o modelo conta com o parâmetro τ^2 que é a variância das observações Y, além dos parâmetros associados a média $\underline{\beta}$.

A inferência para os parâmetros envolvidos no modelo $\underline{\theta} = (\underline{\beta}, \sigma^2, \tau^2\phi)$ pode ser feita baseado na função de verossimilhança, que neste caso toma a forma de uma distribuição Normal Multivariada. Usando resultados padrões da distribuição Normal Multivariada é possível mostrar que $Y \sim NMV(X\underline{beta}, \Sigma)$ onde Σ é dada por:

$$\Sigma = \begin{bmatrix} \sigma^2 + \tau^2 & \sigma^2 \exp(-\frac{u_{12}}{\phi}) \\ \sigma^2 \exp(-\frac{u_{21}}{\phi}) & \sigma^2 + \tau^2 \end{bmatrix}.$$

Com isso, tem-se a função de verossimilhança,

$$L(\underline{\theta}) = (2\pi)^{-\frac{n}{2}} |\Sigma|^{-\frac{1}{2}} \exp^{-\frac{1}{2}(Y - X\underline{\beta})^\top \Sigma^{-1}(Y - X\underline{\beta})}.$$

A função de log-verossimilhança,

$$l(\underline{\theta}) = -\frac{n}{2} log(2\pi) - \frac{1}{2} log|\Sigma| - \frac{1}{2}(Y - X\underline{\beta})^\top \Sigma^{-1}(Y - X\underline{\beta}). \tag{3.1}$$

Nosso objetivo agora é maximizar a função de log-verossimilhança em relação a $\underline{\theta}$. Note que temos dois conjuntos de parâmetros, os associados a média $\underline{\beta}$ e os associados a estrutura de variância e covariância (σ^2, τ^2, ϕ). Note ainda que a log-verossimilhança pode facilmente ser derivada em função de $\underline{\beta}$, já para o caso dos parâmetros que indexam a matriz Σ a derivação não é tão trivial.

Derivando a função 3.1 em relação aos $\underline{\beta}$ e igualando a zero, chegamos aos estimadores de máxima verossimilhança.

$$\hat{\underline{\beta}} = (X^\top \Sigma^{-1} X)^{-1}(X^\top \Sigma^{-1} Y) \tag{3.2}$$

Substituindo a expressão em 3.2 na equação em 3.1 chegamos a função de log-verossimilhança concentrada apenas nos parâmetros que definem a estrutura de variância e covariância do modelo, denote isto por $\underline{\theta}^* = (\sigma^2, \tau^2, \phi)$.

$$l^*(\underline{\theta}^*) = -\frac{n}{2} log(2\pi) - \frac{1}{2} log|\Sigma| - \frac{1}{2}\hat{e}^\top \Sigma^{-1} \hat{e}$$

onde $\hat{e} = (Y - X\hat{\underline{\beta}})$. Os três parâmetros restantes estão dentro da matriz Σ, logo é necessário derivá-los usando cálculo matricial. É possível mostrar que a função escore é dada por,

$$\frac{\partial l^*(\underline{\theta}^*;\underline{Y})}{\partial \theta_i^*} = -\frac{1}{2}Tr\left[\Sigma^{-1}\frac{\partial \Sigma}{\partial \theta_i^*}\right] - \frac{1}{2}\underline{\hat{e}}^\top\left[-\Sigma^{-1}\frac{\partial \Sigma}{\partial \theta_i^*}\Sigma^{-1}\right]\underline{\hat{e}}, \quad i = 1,\ldots,3.$$

onde as matrizes $\dfrac{\partial \Sigma}{\partial \theta_i^*}$ representa a matriz obtida por derivar cada elemento da matriz Σ em relação ao respectivo parâmetro. Para exemplificar, considere o caso de apenas duas observações, a derivada de Σ em relação ao parâmetro σ é

$$\frac{\partial \Sigma}{\partial \sigma} = \begin{bmatrix} 2\sigma & 2\sigma\exp(-\frac{u_{12}}{\phi}) \\ 2\sigma\exp(-\frac{u_{21}}{\phi}) & 2\sigma \end{bmatrix}.$$

Derivando agora em relação ao parâmetro τ,

$$\frac{\partial \Sigma}{\partial \tau} = \begin{bmatrix} 2\tau & 0 \\ 0 & 2\tau \end{bmatrix}.$$

E finalmente em relação ao parâmetro ϕ

$$\frac{\partial \Sigma}{\partial \phi} = \begin{bmatrix} 0 & \sigma^2 u_{12}\exp(-\frac{u_{12}}{\phi})/\phi^2 \\ \sigma^2 u_{21}\exp(-\frac{u_{21}}{\phi})/\phi^2 & 0 \end{bmatrix}.$$

Dado estes resultados estamos aptos a começar a implementação do modelo geoestatístico. Quando trabalhamos com modelos espaciais a parte que requer mais cuidado é a matriz de variância/covariância, então começamos a implementação por ela. A função abaixo chamada *monta.sigma*() monta a estrutura da matriz Σ do modelo geoestatístico.

```
> monta.sigma <- function(s, t, phi, U) {
+     Sigma <- as.matrix(s^2 * exp(-U/phi))
+     diag(Sigma) <- s^2 + t^2
+     return(Sigma)
+ }
```

Sempre que implementamos um modelo complexo é interessante simular dados dele para nos certificarmos que o processo de inferência esta correto. Para a simulação vamos supor que a estrutura de média é composta de apenas um parâmetro $b = 50$, vamos fixar os parâmetros $\sigma^2 = 2$, $\tau^2 = 1$ e $\phi = 0.25$, os dados serão simulados dentro de um quadrado unitário em uma grade regular.

```
> simula.geo <- function(b, s, t, phi, n.simul) {
+     coord.X <- seq(0, 1, l = sqrt(n.simul))
+     coord.Y <- seq(0, 1, l = sqrt(n.simul))
+     grid.simul <- expand.grid(coord.X, coord.Y)
+     U <- dist(grid.simul, diag = TRUE, upper = TRUE)
+     Sigma <- monta.sigma(s = s, t = t, phi = phi, U = U)
+     Z <- rnorm(dim(Sigma)[1])
+     Y = chol(Sigma) %*% Z + b
+     dados <- data.frame(Y = Y, coord.X = grid.simul[, 1], coord.Y = grid.simul[,
+         2])
+     return(dados)
+ }
```

Na função *simula.geo*() começamos criando as coordenadas X e Y e fazendo todas as combinações entre essas, o que gera uma grade regular. Após criar a grade calculamos a matriz de distância entre todos os pontos, chamamos a matriz de U. Com a matriz U e os parâmetros criamos a matriz Σ, simulamos dados de uma Normal independente e aplicamos uma transformação multiplicando o vetor Z pela Cholesky da matriz Σ que induz a dependência entre as observações. Desta forma, temos dados do modelo geoestatístico usando a função de correlação exponencial.

O próximo passo para a inferência é a implementação da função de log-verossimilhança concentrada.

```
> ll.geo <- function(s, t, phi, dados) {
+     U <- dist(dados[, 2:3], diag = TRUE, upper = TRUE)
+     Sigma <- monta.sigma(s = s, t = t, phi = phi, U = U)
+     X <- as.vector(rep(1, l = length(dados$Y)))
+     beta.hat <- solve(t(X) %*% solve(Sigma, X)) %*% (t(X) %*%
+         solve(Sigma, dados$Y))
+     ll = dmvnorm(dados$Y, mean = X %*% beta.hat, sigma = Sigma,
+         log = TRUE)
+     return(-ll)
+ }
```

A partir deste ponto podemos maximizar a log-verossimilhança diretamente através da *optim*() ou qualquer outra forma de maximização numérica. Porém, como obtemos o gradiente analitico vamos utiliza-lo para ajudar no algoritmo de maximização numérica e depois vamos comparar o tempo coputacional com e sem o uso do gradiente analitico.

Para implementar o gradiente precisamos primeiro implementar três funções que implementam cada uma das matrizes de derivadas em relação a cada um dos parâmetros que indexam a matriz Σ.

```
> deriv.s <- function(s, t, phi, U) {
+     Sigma.s <- 2 * s * as.matrix(exp(-U/phi))
+     diag(Sigma.s) <- 2 * s
+     return(Sigma.s)
+ }
> deriv.t <- function(s, t, phi, U) {
+     dimensao <- dim(as.matrix(U))
+     Sigma.t <- matrix(0, ncol = dimensao[1], nrow = dimensao[2])
+     diag(Sigma.t) <- 2 * t
+     return(Sigma.t)
+ }
> deriv.phi <- function(s, t, phi, U) {
+     Sigma.phi <- s^2 * as.matrix(U) * as.matrix(exp(-U/phi))/phi^2
+     return(as.matrix(Sigma.phi))
+ }
```

E agora podemos escrever a função escore completa.

```
> escore <- function(s, t, phi, dados) {
+     U <- dist(dados[, 2:3], diag = TRUE, upper = TRUE)
```

```
+     Sigma <- monta.sigma(s = s, t = t, phi = phi, U = U)
+     X <- as.vector(rep(1, l = length(dados$Y)))
+     beta.hat <- solve(t(X) %*% solve(Sigma, X)) %*% (t(X) %*%
+         solve(Sigma, dados$Y))
+     e.hat <- dados$Y - X %*% beta.hat
+     Sigma1 <- solve(Sigma)
+     de.s <- deriv.s(s = s, t = t, phi = phi, U = U)
+     de.t <- deriv.t(s = s, t = t, phi = phi, U = U)
+     de.phi <- deriv.phi(s = s, t = t, phi = phi, U = U)
+     S1D <- Sigma1 %*% de.s
+     U.s <- -0.5 * sum(diag(S1D)) - 0.5 * (t(e.hat) %*% (-S1D %*%
+         Sigma1) %*% e.hat)
+     T1D <- Sigma1 %*% de.t
+     U.t <- -0.5 * sum(diag(T1D)) - 0.5 * (t(e.hat) %*% (-T1D %*%
+         Sigma1) %*% e.hat)
+     P1D <- Sigma1 %*% de.phi
+     U.phi <- -0.5 * sum(diag(P1D)) - 0.5 * (t(e.hat) %*% (-P1D %*%
+         Sigma1) %*% e.hat)
+     return(-1 * c(U.s, U.t, U.phi))
+ }
```

Um detalhe computacional tanto na função escore como na log-verossimilhança concentrada é que na saída da função pedimos para que seja retornado o negativo da função, isso se deve apenas para compatibilidade com a função $mle2()$ que por default minimiza a função objetivo, no caso a log-verossimilhança.

Com tudo implementado podemos simular um conjunto de dados do modelo e ajustar usando a função $mle2()$ do pacote *bbmle* por conveniência. Poderiamos usar diretamente a função $optim()$ com qualquer um de seus algoritmos, ou mesmo outros maximizadores residentes no R.

A seguinte chamada simula 225 amostras do modelo geoestatístico.

```
> set.seed(12)
> dados <- simula.geo(b = 50, s = 2, t = 1, phi = 0.25, n.simul = 225)
```

Estimando os parâmtros sem o uso do gradiente analitico pelo algoritmo $L-BFGS-B$.

```
> require(bbmle)
> require(mvtnorm)
> system.time(modelo <- mle2(ll.geo, start = list(s = 1, t = 0.1,
+     phi = 0.1), method = "L-BFGS-B", lower = list(s = 1e-32,
+     t = 1e-32, phi = 1e-32), upper = list(s = Inf, t = Inf, phi = Inf),
+     data = list(dados = dados)))

   user  system elapsed
 16.713   0.336  17.051
```

A mesma chamada anterior com o uso do gradiente analitico.

```
> system.time(modelo.escore <- mle2(ll.geo, gr = escore, start = list(s = 1,
+     t = 0.1, phi = 0.1), method = "L-BFGS-B", lower = list(s = 1e-32,
+     t = 1e-32, phi = 1e-32), upper = list(s = Inf, t = Inf, phi = Inf),
+     data = list(dados = dados)))
```

```
   user  system elapsed
  8.944   0.140   9.085
```

De acordo com o tempo computacional exigido por cada chamada podemos verificar que o uso do gradiente analitico diminui o tempo para a maximização da log-verossimilhança praticamente pela metade. O que é um ganho expressivo, principalmente quando trabalha-se com grandes bases de dados. Podemos dar um *summary*() para resumir as informações do ajuste.

```
> summary(modelo.escore)

Maximum likelihood estimation

Call:
mle2(minuslogl = ll.geo, start = list(s = 1, t = 0.1, phi = 0.1),
    method = "L-BFGS-B", data = list(dados = dados), gr = escore,
    lower = list(s = 1e-32, t = 1e-32, phi = 1e-32), upper = list(s = Inf,
        t = Inf, phi = Inf))

Coefficients:
    Estimate Std. Error z value     Pr(z)
s   1.530934   0.192504  7.9527 1.824e-15 ***
t   0.851615   0.260177  3.2732  0.001063 **
phi 0.106833   0.039098  2.7324  0.006287 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

-2 log L: 826.6376
```

Deixamos para o leitor encontrar as estimativas do vetor $\underline{\beta}$, bem como seus respectivos intervalos de confiança. Para finalizar o procedimento de inferência no modelo geoestatístico a Figura 3.1 apresenta os perfis de verossimilhança para os parâmetros que indexam a matriz de variância/covariância do modelo. Note a forte assimetria em todos os parâmetros. O intervalo perfilhado para o parâmetro τ bate na borda esquerda do espaço paramétrico com um nível de confiança de aproximadamente 80%, este tipo de problemas não é incomum em modelos com efeitos espaciais. Pode-se tentar contornar isso fazendo reparametrizações, por exemplo estimar o $log(\tau)$ ou alguma outra função que seja adequada. O importante destacar é que não há uma receita geral que vai funcionar bem todas as vezes, em cada modelo diferentes reparametrizações podem ser tentadas até que se chegue a uma satisfatória.

3.2 Verossimilhança Marginal

Os modelos mistos lineares generalizados são os modelos de regressão com efeitos aleatórios mais comumente usados, quando a variável resposta não é Gaussiana. O objetivo desta Seção é descrever a formulação de um modelo de regressão com efeitos aleatórios de uma forma geral, de tal forma que modelos para dados longitudinais, medidas repetidas, modelos com efeitos espaciais, temporais e espaço temporais possam ser descritos todos no mesmo *framework* unificando toda uma classe de modelos estatísticos que é extremamente poderosa.

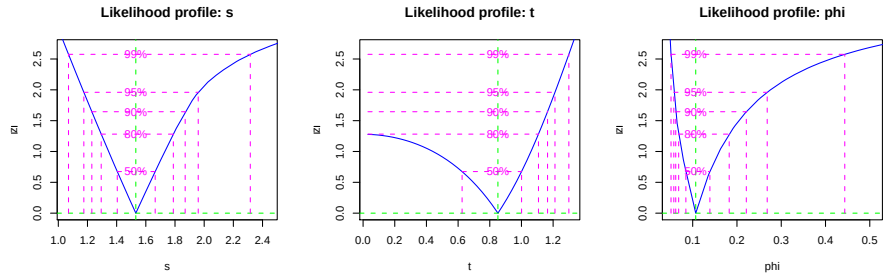

Figura 3.1: Perfis de verossimilhança modelo geoestatístico.

Vamos descrever o modelo no contexto de dados agrupados e conforme necessário vamos indicando pequenas mudanças que geram modelos conhecidos, por exemplo, para dados longitudinais e espaciais. Seja Y_{ij} a j-ésima medida para a unidade amostral $i, i = 1, \ldots, N$, $j = 1, \ldots, n_i$ e $\underline{Y}_i$ o vetor n_i-dimensional de todas as medidas realizadas na unidade amostral i. Assumindo independência condicional no vetor q-dimensional de efeitos aleatórios $\underline{b}_i$, para o qual atribuímos uma distribuição $NMV_q(\underline{0}, \Sigma)$, as respostas Y_{ij} são independentes com densidade da forma,

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

onde $g(\mu_{ij}) = \mathbf{x}_{ij}^T\boldsymbol{\beta} + \mathbf{z}_{ij}^T\mathbf{b_i}$ para uma função de ligação $g(.)$ conhecida, com $\mathbf{x}_{ij}$ e $\mathbf{z}_{ij}$ vetor de covariáveis conhecidas de dimensão p e q respectivamente, $\boldsymbol{\beta}$ um vetor p-dimensional de coeficientes de regressão fixos desconhecidos, e ϕ algum parâmetro extra na verossimilhança, geralmente indicando precisão ou variância. Para completar a especificação do modelo, seja $f(\mathbf{b_i}|\Sigma)$ a densidade da $NMV_q(\mathbf{0}, \Sigma)$ distribuição atribuída para os efeitos aleatórios $\mathbf{b_i}$.

Como já mencionado a estimação dos parâmetros envolvidos no modelo pode ser feita maximizando a verossimilhança marginal, obtida por integrar os efeitos aleatórios. A contribuição para a verossimilhança dada pela unidade amostral i é

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

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

$$L(\underline{\beta}, \Sigma, \phi) = \prod_{i=1}^{N} f_i(\underline{y}_i|\underline{\beta}, \Sigma, \phi) \tag{3.3}$$

$$= \prod_{i=1}^{N} \int \prod_{j=1}^{n_i} f_{ij}(y_{ij}|\underline{b}_i, \underline{\beta}, \phi) f(\underline{b}_i|\Sigma) d\underline{b}_i. \tag{3.4}$$

O problema principal em maximizar 3.3 é a presença das N integrais sobre os efeitos aleatórios q-dimensionais. Em alguns casos especiais estas integrais podem ser resolvidas analiticamente, como no caso do modelo geoestatístico. Porém, na maioria das situações onde a resposta é não Gaussiana as integrais não tem solução analitica.

Além disso, um problema adicional é a dimensão do vetor de efeitos aleatórios, quando q é pequeno o que acontece por exemplo, em modelos de intercepto $q = 1$ ou inclinação aleatório $q = 2$ as integrais são passíveis de ser resolvidas por métodos de integração numérica convencionais, como Gauss-Hermite, Laplace e Monte Carlo que serão detalhadas na sequência. Porém, em modelos espaciais a dimensão do vetor aleatório $q = N$, ou seja, o vetor tem a dimensão do tamanho da amostra possivelmente grande o que torna os métodos de integração numérica, praticamente não aplicáveis. Métodos de integração numérica como Gauss-Hermite e Monte Carlo vão ser utéis quando a dimensão do vetor de efeitos aleatórios é pequena digamos menor que seis. Para efeitos aleatórios de maior dimensão uma implementação muito cuidadosa do método de Laplace pode ser adequada em algumas situações.

Na maioria das vezes quando é necessário trabalhar com modelos onde o vetor de efeitos aleatórios é de grande dimensão o mais usual é trocar o paradigma de Verossimilhança, e passar para o paradigma Bayesiano, onde os métodos MCMC - Monte Carlo via Cadeias de Markov são extremamente poderosos para ajustar modelos de alta complexidade. Na sequência vamos apresentar alguns métodos tradicionais de integração numérica, que podem ser usados quando ajustamos modelos de efeitos aleatórios de baixa complexidade na estrutura aleatória. Os métodos serão aplicadas na estimação de alguns modelos simples para medidas repetidas e dados longitudinais não Gaussianos.

3.3 Técnicas de integração numérica

A necessidade de resolver uma integral numericamente aparece com bastante frequência quando ajustamos modelos de regressão com efeitos aleatórios. Como exemplo didático, escolhemos o modelo Poisson com intercepto aleatório, por ser um modelo simples, conter apenas dois parâmetros o que permite contruir gráficos de contornos da verossimilhança e sua aproximação quadrática. O modelo tem a seguinte forma: $Y_{ij} \sim P(\lambda_{ij})$ onde $ln(\lambda_{ij}) = \beta_0 + b_i$ e $b_i \sim N(0, 1/\tau^2)$. Onde β_0 é o intercepto, b_i o efeito aleatório e τ^2 o parâmetro de precisão. Lembre-se que $i = 1, \ldots, N$ indica o número de unidades amostrais e $j = 1, \ldots, n_i$ indica o número de medidas feitas na unidade amostral i. Neste caso, a contribuição para a verossimilhança de cada unidade amostral é dada por:

$$\begin{aligned} f_i(y_{ij}|b_i, \beta_0) &= \int_{-\infty}^{\infty} \frac{\exp^{-\lambda_{ij}} \lambda_{ij}^{y_{ij}}}{y_{ij}!} \left(\frac{\tau}{2\pi}\right)^{1/2} \exp^{-\frac{\tau^2}{2} b_i^2} db_i \qquad (3.5) \\ &= \int_{-\infty}^{\infty} \frac{\exp^{-\exp(\beta_0+b_i)} \exp^{(\beta_0+b_i)^{y_{ij}}}}{y_{ij}!} \left(\frac{\tau}{2\pi}\right)^{1/2} \exp^{-\frac{\tau^2}{2} b_i^2} db_i. \end{aligned}$$

A integral em 3.5 precisa ser resolvida para cada uma das N unidades amostrais em cada passo de algum algoritmo de maximização numérica. Neste caso, a integral tem apenas uma dimensão, vamos explorar diversos métodos de integração numérica para resolver a integral em 3.5 e posteriormente utiliza-los para estimar os parâmetros envolvidos no modelos de Poisson com intercepto aleatório.

Diversos métodos de integração numérica podem ser encontrados em livros clássicos de cálculo numérico. O método do retângulo, dos trapézios, do ponto

central e suas diversas variações, são métodos simples de serem implementados. Porém, na situação de modelos de regressão com efeitos aleatórios são de pouca valia, por causa da restrição de que a integral a ser resolvidade deve ser própria, ou seja com limites finitos e fixados. O que não é o caso, como podemos ver na equação 3.5. Uma justifiativa para o uso destes métodos é ao invés de integral na reta real, o fazemos em um dominio finito adequado da função integrando, por exemplo, de -10 a 10 sabemos que a grande massa da distribuição deve estar contida, então integramos apenas nesta região da reta.

Dentre os diversos métodos possíveis optamos por descrever o método de Simpson, Quadratura Gaussiana usando os polinômios de Hermite, próprios para a integração na reta real. Métodos baseados em simulação, integração Monte Carlo e Quase Monte Carlo além da aproximação de Laplace. Combinando o método da Quadratura Gaussiana com a aproximação de Laplace, chegamos a Quadratura Adaptativa e o mesmo pode ser feito combinando Quase Monte Carlo com Laplace para obter um Quase Monte Carlo adaptativo.

3.3.1 Simulando de um modelo Poisson com intercepto aleatório

Em todos os métodos de integração numérica que serão apresentados vamos utilizar o modelo de Poisson com intercepto aleatório para exemplificar o cálculo numérico. Para isto, precisamos de amostras deste modelo para podermos avaliar a função de verossimilhança para uma dada configuração de parâmetros, que é o objetivo final do uso dos métodos de integração numérica em modelos de regressão com efeitos aleatórios. A função *simula.poisson*() simula amostras deste modelo de acordo com a parametrização usada.

```
> simula.poisson <- function(X, Z, beta.fixo, prec.pars) {
+     n.bloco <- dim(Z)[2]
+     n.rep <- dim(Z)[1]/n.bloco
+     bi <- rnorm(n.bloco, 0, sd = 1/prec.pars)
+     XZ <- cbind(X, Z)
+     beta <- c(beta.fixo, bi)
+     preditor <- XZ %*% beta
+     lambda <- exp(preditor)
+     y <- rpois(length(lambda), lambda = lambda)
+     return(data.frame(y = y, X = X, ID = rep(1:n.bloco, each = n.rep)))
+ }
```

Para simular do modelo precisamos das matrizes de delineamento X e Z e dos parâmetros β_0 e τ. De acordo com o tamanho das matrizes X e Z a função identifica quantas unidades amostrais e quantas repetições por unidade amostral deve ser simulada. Feita a função podemos usá-la.

```
> ID <- as.factor(rep(1:10, each = 10))
> X <- rep(1, 100)
> Z <- model.matrix(~-1 + ID)
> set.seed(123)
> dados <- simula.poisson(X = X, Z = Z, beta.fixo = 2, prec.pars = 4)
```

De acordo com o código acima, foram simulados 10 unidades amostrais em cada uma destas unidades são retiradas 10 amostras totalizando 100 observações. Neste exemplo, para cada avaliação da verossimilhança devemos resolver 10 integrais uma para cada unidade amostral. Nos exemplos, vamos usar apenas uma unidade amostral e fixar os parâmetros nos valores simulados para avaliar a integral. A função integrando escrita de forma vetorial, fica dada por:

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

Escrever a função em forma vetorial, significa simplesmente que podemos passar em vetor de valores que a função será avaliada em cada um destes valores. Outro fato importante na forma de escrever o integrando é fazer o máximo possível das operações em escala logaritmica. Isso evita muitos erros numéricos, porém devemos sempre ter em mente que estamos calculando a integral na escala original e nunca em logaritmo. Por exemplo,

```
> integrando(b = c(-1, 0, 1), X = X, y = dados[which(dados$ID ==
+     1), ]$y, beta.fixo = 2, prec.pars = 4, log = FALSE)

[1]  0.000000e+00 2.790554e-101  0.000000e+00

> integrando(b = c(-1, 0, 1), X = X, y = dados[which(dados$ID ==
+     1), ]$y, beta.fixo = 2, prec.pars = 4, log = TRUE)

[1] -2024.9364  -231.5349 -2221.6619
```

Tenha sempre em mente o formato vetorial para avaliar a função, esta forma de escrever o integrando facilita muito a construção de algoritmos para integração numérica. É conveniente fazer um gráfico da função integrando para termos uma idéia do formato da função que estamos integrando. A Figura 3.2 apresenta o gráfico do integrando avaliado para cada uma das 10 unidades amostrais simuladas, o eixo y foi padronizada para poder colocar todas as funções no mesmo gráfico.

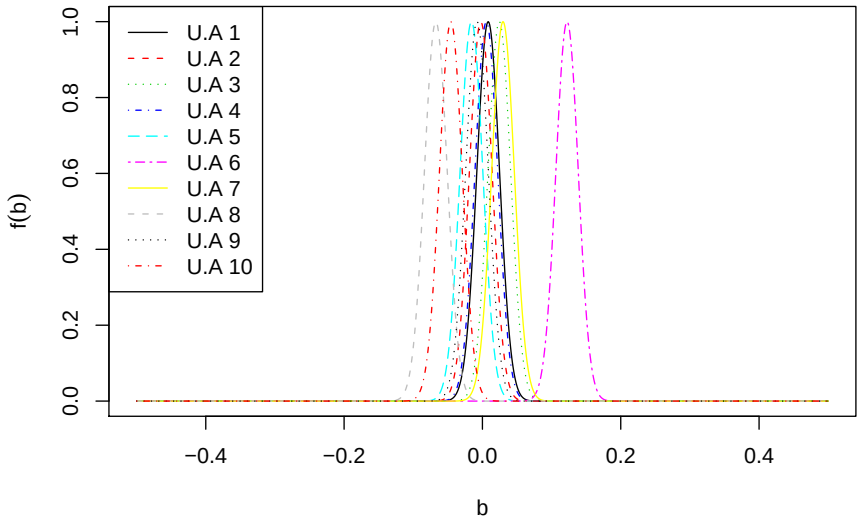

Figura 3.2: Integrando de acordo com unidade amostral - Modelo Poisson com intercepto aleatório.

3.3.2 Método Trapezoidal

O método trapezoidal consiste no uso de uma função linear para aproximar o integrando ao longo do intervalo de integração. O uso do polinômio de Newton entre os pontos $x = a$ e $x = b$ resulta em:

$$f(x) \approx f(a) + (x - a)\left[\frac{f(b) - f(a)}{b - a}\right].$$

Realizando a integração analítica, obtém-se:

$$\begin{aligned} I(f) &\approx \int_a^b f(a) + (x - a)\left[\frac{f(b) - f(a)}{b - a}\right] dx \\ &= f(a)(b - a) + \frac{1}{2}[f(b) - f(a)](b - a) \end{aligned}$$

Simplificando o resultado, obtém-se uma fórmula aproximada popularmente conhecida como regra ou método trapezoidal.

$$I(f) \approx \frac{[f(a) + f(b)]}{2}(b - a) \tag{3.6}$$

A expressão em 3.6 é extremamente simples de ser usada, requer apenas duas avaliações da função integrando. Sua versão R é apresentada abaixo.

```
> trapezio <- function(integrando, a, b, ...) {
+     Int <- ((integrando(a, ...) + integrando(b, ...))/2) * (b -
```

```
+         a)
+     return(Int)
+ }
```

Podemos agora usar o método trapezoidal para avaliar a integral dentro do modelo Poisson com intercepto aleatório.

```
> log(trapezio(integrando = integrando, a = -0.1, b = 0.1, X = X,
+     y = dados[which(dados$ID == 1), ]$y, beta.fixo = 2, prec.pars = 4,
+     log = FALSE))

[1] -249.4511
```

Este método é extremamente simples e serve apenas para apresentar as idéias gerais de integração numérica. Na sequência veremos que o resultado apresentado por este método é muito ruim.

3.3.3 Método de Simpson 1/3

Neste método, um polinômio de segunda ordem é usado para aproximar o integrando. Os coeficientes de um polinômio quadrático podem ser determinadas a partir de três pontos. Para uma integral ao longo do domínio $[a, b]$, são usados os dois pontos finais $x_1 = a$, $x_3 = b$, e o ponto central, $x_2 = (a + b)/2$. O polinômio pode ser escrito na forma:

$$p(x) = \alpha + \beta(x - x_1) + \lambda(x - x_1)(x - x_2) \tag{3.7}$$

onde α, β e λ são constantes desconhecidas avaliadas a partir da condição que diz que o polinômio deve passar por todos os pontos, $p(x_1) = f(x_1)$, $p(x_2) = f(x_2)$ e $p(x_3) = f(x_3)$. Isso resulta em:

$$\alpha = f(x_1), \quad \beta = [f(x_2) - f(x_1)]/(x_2 - x_1) \quad \text{e} \quad \lambda = \frac{f(x_3) - 2f(x_2) + f(x_1)}{2(h)^2}$$

onde $h = (b - a)/2$. Substituindo as constantes de volta em 3.7 e integrando $p(x)$ ao lonfo do intervalo $[a, b]$, obtém-se

$$I = \int_{x_1}^{x_3} f(x)dx \approx \int_{x_1}^{x_3} p(x)dx = \frac{h}{3}\left[f(a) + 4f(\frac{a+b}{2}) + f(b)\right].$$

Note que para o cálculo da integral é necessário apenas três avaliações da função, o que torna o método muito rápido. Podemos também facilmente implementar este método para integrar uma função qualquer, tal função terá como seus argumentos os limites $[a, b]$ e a função a ser integrada.

```
> simpson <- function(integrando, a, b, ...) {
+     h <- (b - a)/2
+     x2 <- (a + b)/2
+     integral <- (h/3) * (integrando(a, ...) + 4 * integrando(x2,
+         ...) + integrando(b, ...))
+     return(integral)
+ }
```

Uma vez implementada a função podemos usá-la para integrar a nossa função de interesse. Lembre-se ainda que para o procedimento de maximização nos interessa o log do valor da integral e não a integral em log, por isso precisamos avaliar a função em sua escala original o que é computacionalmente incoveniente, mas necessário. Além disso, precisamos definir os limites de integração, neste caso fixamos -0.5 a 0.5 tendo em mente o gráfico do integrando. Apenas para comparação dos resultados usamos a função *integrate*() residente no *R*.

```
> simpson(integrando = integrando, a = -0.5, b = 0.5, X = X, y = dados[which(dados$ID
+     1), ]$y, beta.fixo = 2, prec.pars = 4, log = FALSE)

[1] 1.860369e-101

> log(simpson(integrando = integrando, a = -0.5, b = 0.5, X = X,
+     y = dados[which(dados$ID == 1), ]$y, beta.fixo = 2, prec.pars = 4,
+     log = FALSE))

[1] -231.9403

> log(integrate(integrando, lower = -Inf, upper = Inf, X = X, y = dados[which(dados$I
+     1), ]$y, beta.fixo = 2, prec.pars = 4, log = FALSE)$value)

[1] -234.5992
```

O resultado do método de Simpson é compatível com o obtido via *integrate*(), e bastante diferente do obtido pelo método do Trapézio. O mal desempenho do último é basicamente por este estar quase que totalmente voltado aos limites do intervalo de integração, que neste caso são definidos arbitrariamente. Se olharmos para a Figura 3.2 só este massa no intervalo -0.1 a 0.1 se integrarmos a função neste intervalo pelo método do Trapézio chegamos a um valor de -249.4511 mais compatível com os obtidos via Simpson e integrate. O problema que enfrentamos aqui é como definir tais limites em situações práticas de forma geral. E esta é uma das grandes limitações destes métodos, já em uma dimensão, a outra grande limitação é como expandir estes métodos para integrais em maior dimensão e como definir os limites em tais dimensões. Estes problemas, não são de fácil solução e geraram margem para o surgimento de diversos outros métodos que tentam superar tais limitações.

3.3.4 Quadratura de Gauss-Hermite

Nos dois métodos de integração apresentados até agora, a integral de $f(x)$ ao longo do intervalo $[a, b]$ foi avaliada representado $f(x)$ como um polinômio de fácil integração. A integral é avaliada como uma soma ponderada dos valores de $f(x)$ nos diferentes pontos. A localização dos pontos comuns é predeterminada em um dos métodos de integração. Até agora os dois métodos consideram pontos igualmente espaçados. Na quadratura de Gauss, a integral também é avaliada usando uma soma ponderadas dos valores de $f(x)$ em pontos distintos ao longo do intervalo $[a, b]$ (chamados pontos de Gauss). Estes pontos, contudo, não são igualmente espaçados e não incluem os pontos finais. O método de Gauss-Hermite é uma extensão do método de Quadratura Gaussiana para resolver integrais da forma:

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

Neste caso a integral é aproximada por uma soma ponderada, da função avaliada nos pontos de Gauss e pesos de integração.

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

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

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

Para a aproximação de integrais via o método de Gauss-Hermite precisamos dos pesos de integração w_i e dos pontos de Gauss x_i. Em R o pacote *statmod* através da função *gauss.quad*() calcula os pesos e os pontos de Gauss-Hermite. A função abaixo, implementa o método de integração de Gauss-Hermite para uma função qualquer unidimensional.

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

Esta função tem apenas dois argumentos, o primeiro é a função a ser integrada e o segundo o número de pontos a ser utilizado na aproximação. A segunda linha da função faz apenas uma soma ponderada da função avaliada nos pontos de Gauss. O método de Gauss-Hermite apresenta duas grandes limitações. A primeira está relacionada a escolha dos pontos de Gauss, que são escolhidos baseados em e^{-x^2}, independente da função $f(x)$ no integrando. Dependendo do suporte de $f(x)$, os pontos selecionados podem ou não estar dentro da área de interesse. Uma idéia naural é reescalonar os pontos de modo a colocá-los na área de maior densidade da função $f(x)$ o que gera o método chamada de Quadratura Adaptativa de Gauss-Hermite, que veremos adiante. A Figura 3.3 ilustra o problema da colocação dos pontos de integração.

Pela Figura 3.3 fica claro que para integrar a função em preto os pontos ($n = 20$) são satisfatórios, porém para a função em vermelho são claramente inadequados, já que, a área da função de maior massa não tem nenhum ponto de integração. Desta forma, para conseguir um resultado satisfatório é necessário aumentar muito o número de pontos de integração, encarecendo o procedimento. Vamos usar esta função para avaliar o valor da integral, contida no modelo Poisson com intercepto aleatório.

```
> log(gauss.hermite(integrando = integrando, n.pontos = 21, X = X,
+     y = dados[which(dados$ID == 1), ]$y, beta.fixo = 2, prec.pars = 4,
+     log = FALSE))

[1] -232.2709
```

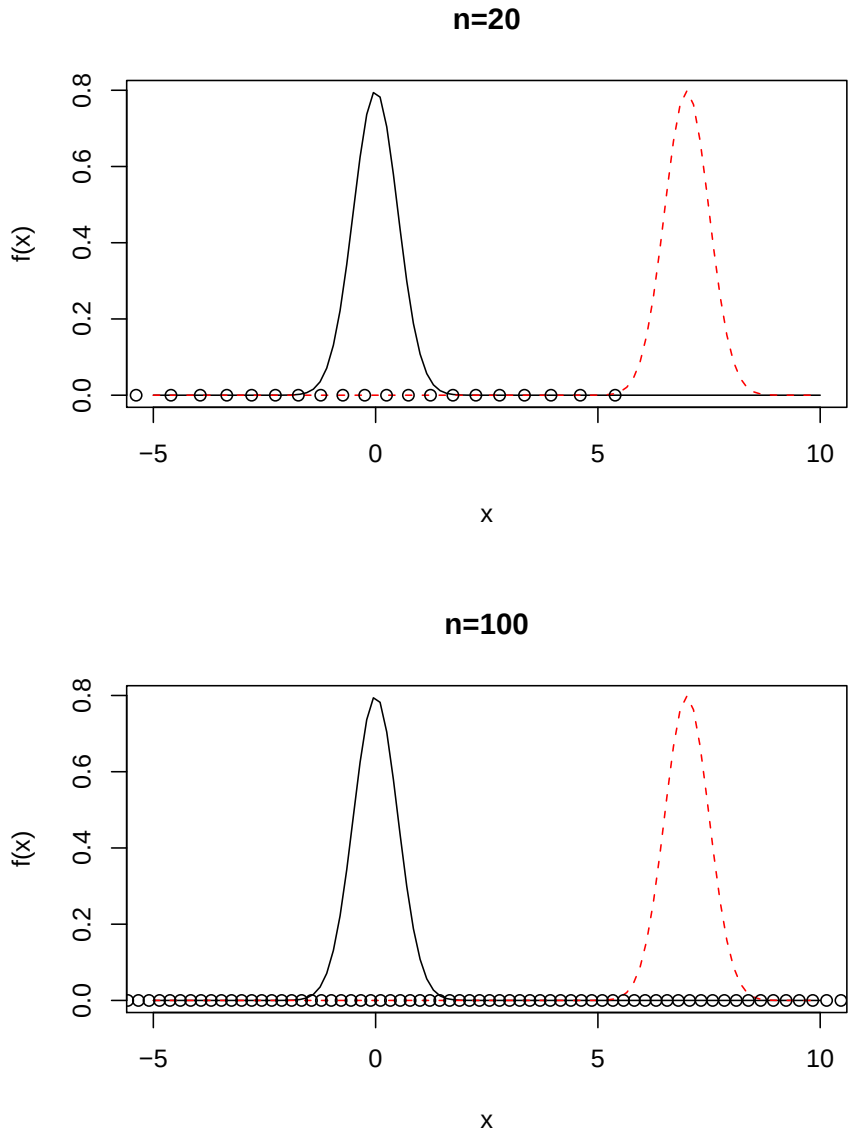

Figura 3.3: Espalhamento dos pontos de integração pelo método de Gauss-Hermite.

O segundo problema do método de Gauss-Hermite está relacionado com a dimensão da integral a ser resolvida. Quando a função é unidimensional, basta espalhar os pontos sobre a reta real e avaliar a função neste pontos. Para funções em duas ou mais dimensões precisamos do produto cartesiano dos pontos de integração para espalhar na função multidimensional, ou seja, o número de pontos cresce exponencialmente de acordo com a dimensão da função a ser integrada. Por exemplo, se em uma dimensão usamos 20 pontos para a integração em duas dimensões precisamos de $20^2 = 400$, em três $20^3 = 8000$. Isso mostra para integrar funções multidimensionais o método de Gauss-Hermite torna rapidamente proibitivo. O método de Quadratura Adaptativa de Gauss-Hermite ameniza um pouco este problema, por precisar de menos pontos de integração, porém o problema persiste para dimensões maiores que cinco ou seis em geral.

A função abaixo implementa o método de Gauss-Hermite para dimensões maiores que um.

```
> gauss.hermite.multi <- function(integrando, n.dim, n.pontos,
+     ...) {
+     normaliza <- function(x) {
+         exp(-t(as.numeric(x)) %*% as.numeric(x))
+     }
+     pontos <- gauss.quad(n.pontos, kind = "hermite")
+     nodes <- matrix(rep(pontos$nodes, n.dim), ncol = n.dim)
+     pesos <- matrix(rep(pontos$weights, n.dim), ncol = n.dim)
+     lista.nodes <- list()
+     lista.pesos <- list()
+     for (i in 1:ncol(nodes)) {
+         lista.nodes[[i]] <- nodes[, i]
+         lista.pesos[[i]] <- pesos[, i]
+     }
+     nodes = as.matrix(do.call(expand.grid, lista.nodes))
+     pesos = do.call(expand.grid, lista.pesos)
+     pesos.grid = apply(pesos, 1, prod)
+     norma = apply(nodes, 1, normaliza)
+     integral <- sum(pesos.grid * (integrando(nodes, ...)/norma))
+     return(integral)
+ }
```

Vamos usar a função *gauss.hermite.multi*() em uma dimensão apenas para exemplificar sua chamada.

```
> log(gauss.hermite.multi(integrando = integrando, n.pontos = 21,
+     n.dim = 1, X = X, y = dados[which(dados$ID == 1), ]$y, beta.fixo = 2,
+     prec.pars = 4, log = FALSE))

[1] -232.2709
```

3.3.5 Adaptativa Gauss-Hermite e Aproximação de Laplace

Com adaptativa Gauss-Hermite, os pontos de integração serão centrados e escalonados como se $f(x)e^{-x^2}$ fosse a distribuição Normal. A média deste Normal será

a moda $\hat{x}$ de $ln[f(x)e^{-x^2}]$, e a variância será igual

$$\left[-\frac{\partial^2}{\partial x^2} ln[f(x)e^{-x^2}]|_{z=\hat{z}}\right]^{-1}$$

Assim, o novos pontos de integração adaptados serão dados por

$$x_i^+ = \hat{x} + \left[-\frac{\partial^2}{\partial x^2} ln[f(x)e^{-x^2}]|_{z=\hat{z}}\right]^{-1/2} x_i$$

com correspondentes pesos

$$w_i^+ = \left[-\frac{\partial^2}{\partial x^2} ln[f(x)e^{-x^2}]|_{z=\hat{z}}\right]^{-1/2} \frac{e^{x_i^+}}{e^{-x_i}} w_i.$$

Como antes a integral é agora aproximada por

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

Note que, quando Gauss-Hermite ou adaptativa Gauss-Hermite é usada no ajuste de modelos de regressão com efeitos aleatórios, uma aproximação é aplicada para a contribuição na verossimilhança para cada uma das N unidades amostrais no conjunto de dados. Em geral, quanto maior a ordem de n pontos de integração melhor será a aproximação. Tipicamente, adaptativa Gauss-Hermite precisa de muito menos pontos que Gauss-Hermite. Por outro lado, adaptativa Gauss-Hermite requer o cálculo de $\hat{x}$ para cada unidade amostral no conjunto de dados, assim a maximização numérica do integrando encarece bastante o custo computacional desta abordagem. Além disso, como o integrando é função dos parâmetros desconhecidos $\underline{\beta}$,Σ e ϕ, os pontos de quadratura, bem como os pesos usados na adaptativa Gauss-Hermite dependem destes parâmetros, e assim precisam ser atualizados a cada passo de um processo de estimação iterativo, através de algum maximizador numérico, como os encontrados na *optim*().

Um caso espacial ocorre quando adaptativa Gauss-Hermite é aplicado com um ponto de integração. Denote $f(x)e^{-x^2}$ por $Q(x)$. Como $n = 1$, $x_1 = 0$ e $w_1 = 1$, nos obtemos $x_1^+ = \hat{x}$, que é o máximo de $Q(x)$. Além disso, os pesos de integração são iguais a

$$w_1^+ = |Q''(\hat{x})|^{-1/2} \frac{e^{-\hat{x}}}{e^{-0}} = (2\pi)^{n/2} |Q''(\hat{x})|^{-1/2} \frac{e^{Q(\hat{x})}}{f(\hat{x}}.$$

Assim, a aproximação fica dada por

$$\begin{aligned} \int f(x)e^{-x^2} dx &= \int e^{Q(x)} dx \\ &\approx w_1^+ f(x_1^+) = (2\pi)^{n/2} |Q''(\hat{x})|^{-1/2} e^{Q(\hat{x})}, \end{aligned}$$

mostrando que a adaptativa Gauss-Hermite com um ponto de integração é equivalente a aproximar o integrando usando a Aproximação de Laplace. A função *laplace*() abaixo implementa a aproximação de Laplace para uma função qualquer.

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

Note a necessidade do uso da *optim*() para encontrar a moda da função e obter o Hessiano numérico. Importante, notar que na chamada para a integração via aproximação de Laplace a função integrando deve estar em escala logaritmica. A chamada abaixo, exemplifica o usa desta aproximação.

```
> log(laplace(integrando, otimizador = "BFGS", n.dim = 1, X = X,
+     y = dados[which(dados$ID == 1), ]$y, beta.fixo = 2, prec.pars = 4,
+     log = TRUE))
```

```
[1] -234.5976
attr(,"logarithm")
[1] TRUE
```

Para finalizar com o uso de integração por Quadratura, a função *adaptative.gauss.hermite*() implementa a integração adaptativa de Gauss-Hermite para uma função qualquer.

```
> adaptative.gauss.hermite <- function(funcao, n.dim, n.pontos,
+     otimizador, ...) {
+     normaliza <- function(x) {
+         exp(-t(as.numeric(x)) %*% as.numeric(x))
+     }
+     pontos <- gauss.quad(n.pontos, kind = "hermite")
+     integral <- -999999
+     inicial <- rep(0, n.dim)
+     temp <- try(optim(inicial, funcao, ..., method = otimizador,
+         hessian = TRUE, control = list(fnscale = -1)))
+     z.chapeu <- temp$par
+     sd.chapeu <- sqrt(diag(solve(-temp$hessian)))
+     mat.nodes <- matrix(NA, ncol = n.dim, nrow = n.pontos)
+     mat.pesos <- matrix(NA, ncol = n.dim, nrow = n.pontos)
+     for (i in 1:length(z.chapeu)) {
+         mat.nodes[, i] <- z.chapeu[i] + sd.chapeu[i] * pontos$nodes
+         mat.pesos[, i] <- sd.chapeu[i] * (exp(-mat.nodes[, i]^2)/exp(-pontos$nodes^2)) *
+             pontos$weights
+     }
+     lista.nodes <- list()
+     lista.pesos <- list()
+     for (i in 1:ncol(mat.nodes)) {
```

```
+         lista.nodes[[i]] <- mat.nodes[, i]
+         lista.pesos[[i]] <- mat.pesos[, i]
+     }
+     nodes = as.matrix(do.call(expand.grid, lista.nodes))
+     pesos = do.call(expand.grid, lista.pesos)
+     pesos.grid = apply(pesos, 1, prod)
+     norma = apply(nodes, 1, normaliza)
+     integral <- sum(pesos.grid * (exp(funcao(nodes, ...))/norma))
+     return(integral)
+ }
```

Para comparar os resultados utilizamos a função usando diferentes quantidade de pontos de integração.

```
> log(adaptative.gauss.hermite(integrando, otimizador = "BFGS",
+     n.dim = 1, n.pontos = 1, X = X, y = dados[which(dados$ID ==
+         1), ]$y, beta.fixo = 2, prec.pars = 4, log = TRUE))

[1] -234.9442

> log(adaptative.gauss.hermite(integrando, otimizador = "BFGS",
+     n.dim = 1, n.pontos = 10, X = X, y = dados[which(dados$ID ==
+         1), ]$y, beta.fixo = 2, prec.pars = 4, log = TRUE))

[1] -234.5976

> log(adaptative.gauss.hermite(integrando, otimizador = "BFGS",
+     n.dim = 1, n.pontos = 21, X = X, y = dados[which(dados$ID ==
+         1), ]$y, beta.fixo = 2, prec.pars = 4, log = TRUE))

[1] -234.5976
```

Com isso, terminamos nossa explanação dos métodos baseados na idéia de aproximar o integrando por algum tipo de polinômio que seja de fácil integração, e usar este como uma aproximação para a verdadeira integral. Na sequência, vamos apresentar um método diferente baseado em simulação, a idéia implicita é estimar o valor da integral. Este procedimento recebe o nome de integração Monte Carlo, além do método básico vamos apresentar algumas variações como o método de Quase Monte Carlo e Quase Monte Carlo adaptativo.

3.3.6 Integração Monte Carlo

Integração Monte Carlo é um simples e poderoso método para aproximar integrais complicadas. Assuma que desejamos estimar o valor da integral de uma função $f(x)$ em algum dominio D qualquer, ou seja,

$$I = \int_D f(x)dx \tag{3.8}$$

A função não precisa ser unidimensional. De fato, técnicas Monte Carlo são muito usadas para resolver integrais de alta dimensão, além de integrais que não tem solução analitica, como no nosso caso.

Assuma que nos temos uma função densidade de probabilidade $p(x)$ cujo dominio coincide com o dominio D. Então, a integral em 3.8 é equivalente a

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

Essa integral nada mais é que a $E\left(\frac{f(x)}{p(x)}\right)$, ou seja, o valor esperado de $\frac{f(x)}{p(x)}$ com respeito a variável aleatória distribuída como $p(x)$. Esta igualdade é verdadeira para qualquer função densidade de probabilidade em D, desde que $p(x) \neq 0$ sempre que $f(x) \neq 0$.

Pode-se estimar o valor de $E\left(\frac{f(x)}{p(x)}\right)$ gerando número aleatórios de acordo com $p(x)$, calcular $f(x)/p(x)$ para cada amostra, e calcular a média destes valores. Quanto mais amostras forem geradas, esta média converge para o verdadeiro valor da integral. Em resumo, para avaliar a integral da função $f(x)$ usando integração Monte Carlo , nos geramos amostras aleatórias de acordo com $p(x)$, e calculamos a média de $f(x)/p(x)$ para todas as amostras.

No caso especifico de modelos de regressão com efeitos aleatórias, a grande maioria das integrais devem ser resolvidas nos reais, ou seja, precisamos de uma distribuição com este suporte para usar como $p(x)$. A escolha mais natural é uma distribuição Normal Multivariada dependendo da dimensão da integral a ser resolvida. Além disso, precisamos decidir a parametrização desta distribuição, ou seja, qual será seu vetor de média e sua matriz de variância/covariância. Em algoritmos básicos o mais comum é usar o vetor de média como 0 e variância unitária. Mas, podemos adaptar este método de forma muito similar ao adaptative Gauss-Hermite, de como espalhar os pontos pelo integrando de forma a cobrir melhor a área do integrando.

Além do espalhamento dos pontos, a geração dos pontos aleatórios também é um fator importante para este método. Como números aleatórios serão gerados cada rodada do algoritmo vai resultar em diferentes valores para a integral o que é indesejável dentro de maximizadores numéricos. Uma abordagem ligeiramente diferente são os métodos Quase Monte Carlo, que diferem apenas na forma de geração dos numéros que passam a ser pseudo-aleatórios gerados de acordo com uma sequência de baixa discrepância. Duas opções para a geração destas sequência de baixa discrepancia, estão disponíveis no pacote *fOptions*, são elas *Halton* e *Sobol* o mecanismo é exatamente o mesmo, só muda que ao invés de usar os números gerados, por exemplo, pela função *rnorm*() usamos estas sequencias de baixa discrepância.

Para exemplificar a ideia de integração Monte Carlo e Quase Monte Carlo, a função *monte.carlo*() implementa o método para uma função qualquer, e permite ao usuário escolher a forma de espalhamento dos pontos.

```
> monte.carlo <- function(funcao, n.dim, n.pontos, tipo, ...) {
+     if (tipo == "MC") {
+         pontos <- rmvnorm(n.pontos, mean = rep(0, n.dim))
+     }
+     if (tipo == "Halton") {
+         pontos <- rnorm.halton(n.pontos, n.dim)
+     }
+     if (tipo == "Sobol") {
```

```
+         pontos <- rnorm.sobol(n.pontos, n.dim)
+     }
+     norma <- apply(pontos, 1, dmvnorm)
+     integral <- mean(funcao(pontos, ...)/norma)
+     return(integral)
+ }
```

Como em todos os métodos apresentados até aqui, vamos usar a função *monte.carlo*() para resolver a integral contida no modelo Poisson com intercepto aleatório.

```
> log(monte.carlo(integrando, n.dim = 1, tipo = "MC", n.pontos = 20,
+     X = X, y = dados[which(dados$ID == 1), ]$y, beta.fixo = 2,
+     prec.pars = 4, log = FALSE))

[1] -233.4942

> log(monte.carlo(integrando, n.dim = 1, tipo = "Halton", n.pontos = 20,
+     X = X, y = dados[which(dados$ID == 1), ]$y, beta.fixo = 2,
+     prec.pars = 4, log = FALSE))

[1] -233.6115

> log(monte.carlo(integrando, n.dim = 1, tipo = "Sobol", n.pontos = 20,
+     X = X, y = dados[which(dados$ID == 1), ]$y, beta.fixo = 2,
+     prec.pars = 4, log = FALSE))

[1] -233.6116
```

O mesmo problema na forma de espalhamento dos pontos encontrados no método de Quadratura de Gauss-Hermite, vale nos métodos de Monte Carlo e Quase Monte Carlo, note que os pontos são sorteados de uma Normal de média 0 e variância 1, quando o integrando não for adequadamente coberto por estes pontos a integração será ruim. Podemos novamente adaptar os pontos de interação que agora são as amostras sorteadas, de modo a explorar melhor o integrando, espalhando os pontos em volta de sua moda de acordo como Hessiano obtido no ponto modal. O processo de adequação dos pontos é identico ao da adaptativa Gauss-Hermite, e não será detalhado novamente aqui. A função *adaptative.monte.carlo*() implementa este método para uma função qualquer.

```
> adaptative.monte.carlo <- function(funcao, n.pontos, n.dim, tipo,
+     otimizador, ...) {
+     if (tipo == "MC") {
+         pontos <- rmvnorm(n.pontos, mean = rep(0, n.dim))
+     }
+     if (tipo == "Halton") {
+         pontos <- rnorm.halton(n.pontos, n.dim)
+     }
+     if (tipo == "Sobol") {
+         pontos <- rnorm.sobol(n.pontos, n.dim)
+     }
+     integral <- -999999
```

```
+     inicial <- rep(0, n.dim)
+     temp <- try(optim(inicial, funcao, ..., method = otimizador,
+         hessian = TRUE, control = list(fnscale = -1)))
+     if (class(temp) != "try-error") {
+         z.chapeu <- temp$par
+         H <- solve(-temp$hessian)
+         sd.chapeu <- sqrt(diag(H))
+         mat.nodes <- matrix(NA, ncol = n.dim, nrow = n.pontos)
+         for (i in 1:length(z.chapeu)) {
+             mat.nodes[, i] <- z.chapeu[i] + sd.chapeu[i] * pontos[,
+                 i]
+         }
+         norma <- dmvnorm(mat.nodes, mean = z.chapeu, sigma = H,
+             log = TRUE)
+         integral = mean(exp(funcao(mat.nodes, ...) - norma))
+     }
+     return(integral)
+ }
```

Novamente, vamos usar a função *adaptative.monte.carlo*() para resolver a integral contida no modelo Poisson com intercepto aleatório.

```
> log(adaptative.monte.carlo(integrando, n.dim = 1, tipo = "MC",
+     n.pontos = 20, otimizador = "BFGS", X = X, y = dados[which(dados$ID ==
+         1), ]$y, beta.fixo = 2, prec.pars = 4, log = TRUE))

[1] -234.5977

> log(adaptative.monte.carlo(integrando, n.dim = 1, tipo = "Halton",
+     n.pontos = 20, otimizador = "BFGS", X = X, y = dados[which(dados$ID ==
+         1), ]$y, beta.fixo = 2, prec.pars = 4, log = TRUE))

[1] -234.5974

> log(adaptative.monte.carlo(integrando, n.dim = 1, tipo = "Sobol",
+     n.pontos = 20, otimizador = "BFGS", X = X, y = dados[which(dados$ID ==
+         1), ]$y, beta.fixo = 2, prec.pars = 4, log = TRUE))

[1] -234.5975
```

Nesta Seção, apresentamos diversos métodos de integração numérica. Na sequência veremos alguns exemplos de modelos de regressão com efeitos aletórios e como usar estes diversos métodos para a estimação por Máxima Verossimilhança dentro deste contexto.

3.4 Modelo Poisson com Intercepto aleatório

Vimos na Seção 3.3.1 como simular de um modelo Poisson com intercepto aleatório. Agora vamos ver como usar os diversos métodos de integração numérica dentro do processo de estimação dos parâmetros deste modelo que são $\underline{\theta} = (\beta_0, \tau)$. O primeiro passo é escrever o modelo completo.

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

Vamos montar uma função generica que capta um conjunto de dados e monta a log-verossimilhança, já integrada de acordo com as variáveis opções de integração apresentadas anteriormente. Essa função vai ser usadas em todos os modelos com efeitos aleatórios apresentados neste texto.

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

```
+         }
+         if (integral == "QMS") {
+             ll[i] <- monte.carlo(modelo, n.pontos = pontos, n.dim = n.dim,
+                 tipo = "Sobol", X = X, Z = Z, Y = dados.id[[i]]$y,
+                 beta.fixo = beta.fixo, prec.pars = prec.pars,
+                 log = FALSE)
+         }
+         if (integral == "AGH") {
+             ll[i] <- adaptative.gauss.hermite(modelo, n.pontos = pontos,
+                 n.dim = n.dim, otimizador = otimizador, X = X,
+                 Z = Z, Y = dados.id[[i]]$y, beta.fixo = beta.fixo,
+                 prec.pars = prec.pars, log = TRUE)
+         }
+         if (integral == "AMC") {
+             ll[i] <- adaptative.monte.carlo(modelo, n.pontos = pontos,
+                 n.dim = n.dim, otimizador = otimizador, tipo = "MC",
+                 X = X, Z = Z, Y = dados.id[[i]]$y, beta.fixo = beta.fixo,
+                 prec.pars = prec.pars, log = TRUE)
+         }
+         if (integral == "AQMH") {
+             ll[i] <- adaptative.monte.carlo(modelo, n.pontos = pontos,
+                 n.dim = n.dim, otimizador = otimizador, tipo = "Halton",
+                 X = X, Z = Z, Y = dados.id[[i]]$y, beta.fixo = beta.fixo,
+                 prec.pars = prec.pars, log = TRUE)
+         }
+         if (integral == "AQMS") {
+             ll[i] <- adaptative.monte.carlo(modelo, n.pontos = pontos,
+                 n.dim = n.dim, otimizador = otimizador, tipo = "Sobol",
+                 X = X, Z = Z, Y = dados.id[[i]]$y, beta.fixo = beta.fixo,
+                 prec.pars = prec.pars, log = TRUE)
+         }
+     }
+     return(sum(log(ll)))
+ }
```

A função *verossimilhanca*() é genérica, vamos mostrar como usá-la no contexto do modelo Poisson com intercepto aleatório.

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

Usando a função $mle2()$ para estimar os parâmetros via o algoritmo $BFGS$, e aproximação de Laplace temos o seguinte.

```
> P.laplace = mle2(mod.Poisson, start = list(b0 = 0, tau = log(1/4)),
+     data = list(integral = "LAPLACE", otimizador = "BFGS", n.dim = 1,
```

```
+         dados = dados))
> summary(P.laplace)

Maximum likelihood estimation

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

Coefficients:
    Estimate Std. Error z value     Pr(z)
b0  2.006958   0.072709 27.6027 < 2.2e-16 ***
tau 1.620670   0.290304  5.5827 2.369e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

-2 log L: 491.6252
```

Vamos avaliar o valor da log-verossimilhança neste pontos pelos outros métodos.

```
> par <- coef(P.laplace)
> ll.Laplace <- mod.Poisson(b0 = par[1], tau = par[2], integral = "LAPLACE",
+     pontos = 21, n.dim = 1, otimizador = "BFGS", dados = dados)
> ll.GH <- mod.Poisson(b0 = par[1], tau = par[2], integral = "GH",
+     pontos = 21, n.dim = 1, dados = dados)
> ll.MC <- mod.Poisson(b0 = par[1], tau = par[2], integral = "MC",
+     pontos = 21, n.dim = 1, dados = dados)
> ll.QMH <- mod.Poisson(b0 = par[1], tau = par[2], integral = "QMH",
+     pontos = 21, n.dim = 1, dados = dados)
> ll.QMS <- mod.Poisson(b0 = par[1], tau = par[2], integral = "QMS",
+     pontos = 21, n.dim = 1, dados = dados)
> ll.AGH <- mod.Poisson(b0 = par[1], tau = par[2], integral = "AGH",
+     pontos = 21, n.dim = 1, otimizador = "BFGS", dados = dados)
> ll.AMC <- mod.Poisson(b0 = par[1], tau = par[2], integral = "AMC",
+     pontos = 21, n.dim = 1, otimizador = "BFGS", dados = dados)
> ll.AQMH <- mod.Poisson(b0 = par[1], tau = par[2], integral = "AQMH",
+     pontos = 21, n.dim = 1, otimizador = "BFGS", dados = dados)
> ll.AQMS <- mod.Poisson(b0 = par[1], tau = par[2], integral = "AQMS",
+     pontos = 21, n.dim = 1, otimizador = "BFGS", dados = dados)
> c(ll.Laplace, ll.GH, ll.MC, ll.QMH, ll.QMS, ll.AGH, ll.AMC, ll.AQMH,
+     ll.AQMS)

[1] 245.8126 243.9302 248.7211 245.2455 244.9611 245.8104 245.8128 245.7723
[9] 245.8448
```

Apesar de todos os métodos apresentarem valores muito proximos do obtido pela aproximação de Laplace, não significa que todos se comportam de forma igual, dentro dos maximizadores numéricos. Por exemplo, o método Monte Carlo, precisa de muitos pontos para conseguir convergência, o que o torna

muito lento nesta forma de abordagem ingenua. Por exemplo, a cada iteração estamos resorteando de uma Normal Multivariada o que demora um tempo razoavel, enquanto que isso não é necessário, podemos sortear apenas uma vez e usar os mesmos pontos em todas as interações do algoritmo numérico. O mesmo se aplicada a todos os outros métodos, porém com esta abordagem não é possível escrever uma função autocontida como a apresentada apenas para fins didáticos. Num algoritmo mais eficiente basta passar os pontos de integração como argumento da função e retirar toda a parte em que os pontos são criados, o mesmo vale para a Quadratura de Gauss-Hermite. Ressaltamos que neste momento não estamos interessados em eficiência computacional, apenas em apresentar os aspectos gerais do métodos de integração numérica. O mesmo ajuste usando quadratura de Gauss-Hermite.

```
> P.GH = mle2(mod.Poisson, start = list(b0 = 0, tau = log(1/4)),
+     data = list(integral = "GH", pontos = 100, n.dim = 1, dados = dados))
> summary(P.GH)

Maximum likelihood estimation

Call:
mle2(minuslogl = mod.Poisson, start = list(b0 = 0, tau = log(1/4)),
    data = list(integral = "GH", pontos = 100, n.dim = 1, dados = dados))

Coefficients:
    Estimate Std. Error z value     Pr(z)
b0  2.016415   0.073391 27.4749 < 2.2e-16 ***
tau 1.635619   0.291663  5.6079 2.048e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

-2 log L: 491.7102

> P.AQMS = mle2(mod.Poisson, start = list(b0 = 0, tau = log(1/4)),
+     data = list(integral = "AQMS", pontos = 10, otimizador = "BFGS",
+         n.dim = 1, dados = dados))
> summary(P.AQMS)

Maximum likelihood estimation

Call:
mle2(minuslogl = mod.Poisson, start = list(b0 = 0, tau = log(1/4)),
    data = list(integral = "AQMS", pontos = 10, otimizador = "BFGS",
        n.dim = 1, dados = dados))

Coefficients:
    Estimate Std. Error z value     Pr(z)
b0  2.006742   0.072547 27.6613 < 2.2e-16 ***
tau 1.622157   0.287350  5.6452 1.650e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

-2 log L: 491.7037
```

Com isso, passamos por todas as etapas do processo de estimação de um modelo de regressão com efeitos aleatórios. Na sequência, vamos mostrar alguns modelos mais complicados, com diferentes estruturas de efeitos aleatórios, porém os métodos de integração serão exatamente os mesmos, bem como todo o processo de construção do modelo. O que irá mudar é apenas a matriz Z que delinea os efeitos aleatórios.