FBST seqüencial

Marcelo Leme de Arruda

TESE APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO TÍTULO
DE
DOUTOR EM CIÊNCIAS

Programa: **Estatística**
Orientador: **Prof. Dr. Sergio Wechsler**

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro da CAPES e da CNPq

São Paulo, junho de 2012

FBST seqüencial

Esta versão da tese contém as correções e alterações sugeridas pela Comissão Julgadora durante a defesa da versão original do trabalho, realizada em 04/06/2012. Uma cópia da versão original está disponível no Instituto de Matemática e Estatística da Universidade de São Paulo.

Comissão Julgadora:

- Prof. Dr. Sergio Wechsler (orientador) - IME-USP
- Prof. Dr. Carlos Alberto de Bragança Pereira - IME-USP
- Prof. Dr. Marcelo de Souza Lauretto - IME-USP
- Prof. Dr. Caio Lucidius Naberezny Azevedo - UNICAMP
- Prof. Dr. Helio dos Santos Migon - UFRJ

Agradecimentos

A meus pais, Ruth e José Paulo, a meus irmãos Ana Maria e Daniel e à minha cunhada Lilian pelo apoio incondicional ao longo dessa caminhada.

À minha avó, aos meus tios, primos e amigos, à Dra. Sylvia, ao Celso e à Coca por, cada qual da sua forma, terem me incentivado durante todos esses anos.

Ao Sergio Wechsler, meu amigo e orientador desde os tempos do mestrado (ainda no século passado) por tudo, especialmente pelas infinitas competência e paciência durante a gestação desta tese; e ao também meu amigo Luiz Gustavo Esteves, por aceitar desde o começo ser meu "co-orientador informal" e pelas muitas opiniões, indicações e manifestações importantes ao longo desse processo.

Aos professores Marcelo Lauretto, Caio Azevedo e Helio Migon e, especialmente, ao professores Carlos Alberto Pereira e Julio Stern, pais do FBST, pela honra de tê-los presentes à minha defesa.

Ao Silvio Rodrigues e à Patrícia Klarmann por suas teses, de grande importância na confecção desta; e ao Rafael Grisi, ao Victor Fossaluza, ao Luís Eduardo Montoya-Delgado, aos ex-funcionários da CPG Pinho e Leka, aos professores Adilson Simonis, Fábio Machado, Yoshiharu Kohayakawa, Carlos Alberto Pereira (novamente), Ernesto Birgin e Regina Madruga: cada qual de sua forma, cada um ao seu tempo, todos foram muito importante para a superação de dificuldades que surgiram ao longo desses anos.

À querida Tia Lucy, por cujas mãos dei os primeiros passos na minha jornada acadêmica, e que nos deixou quando a tese entrava em sua reta final. De onde agora está, ela certamente está olhando por todos os seus "aluninhos" e, com certeza, intercedeu pelo meu sucesso nessa caminhada.

Resumo

ARRUDA, M. L. **FBST Seqüencial**. 2012. 110f. Tese (Doutorado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2012.

O FBST (*Full Bayesian Significance Test*) é um instrumento desenvolvido por Pereira e Stern (1999) com o objetivo de apresentar uma alternativa bayesiana aos testes de hipóteses precisas. Desde sua introdução, o FBST se mostrou uma ferramenta muito útil para a solução de problemas para os quais não havia soluções freqüentistas. Esse teste, contudo, depende de que a amostra seja coletada uma única vez, após o que a distribuição *a posteriori* dos parâmetros é obtida e a medida de evidência, calculada

Ensejadas por esse aspecto, são apresentadas abordagens analíticas e computacionais para a extensão do FBST ao contexto de decisão seqüencial (DeGroot, 2004). É apresentado e analisado um algoritmo para a execução do FBST Seqüencial, bem como o código-fonte de um *software* baseado nesse algoritmo.

Palavras-chave: estatística bayesiana, FBST, teoria de decisão

Abstract

ARRUDA, M. L. **Sequential FBST**. 2012. 110f. Tese (Doutorado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2012.

FBST (*Full Bayesian Significance Test*) is a tool developed by Pereira and Stern (1999), to show a bayesian alternative to the tests of precise hypotheses. Since its introduction, FBST has shown to be a very useful tool to solve problems to which there were no frequentist solutions. This test, however, needs that the sample be collected just one time and, after this, the parameters' *posterior* distribution is obtained and the evidence measure, computed.

Suggested by this feature, there are presented analytic and computational approaches to the extension of the FBST to the sequential decision context (DeGroot, 2004). It is presented and analyzed an algorithm to execute the Sequential FBST, as well as the source code of a software based on this algorithm.

Keywords: bayesian statistic, FBST, decision theory

Sumário

Capítulo 1

Introdução

O FBST (*Full Bayesian Significance Test*) foi desenvolvido por Pereira e Stern (1999) com o objetivo de apresentar uma nova resposta bayesiana (além do então já existente Teste de Jeffreys (1961)) a muitas discussões, tanto técnicas quanto filosóficas, acerca dos testes de hipóteses precisas. Entre outros aspectos, tais discussões abordam:

- a plausibilidade (ou não) de se considerar uma hipótese precisa;
- a característica *ad hoc* da *priori* em H_0 do Teste de Jeffreys (1961), quando o espaço paramétrico Θ é contínuo;
- a probabilidade (e sua plausibilidade - ou não) $P(H_0)$;
- as distinções e semelhanças entre testes de significância e testes de hipóteses
- etc.

Esse teste se fundamenta em uma medida bayesiana de evidência, alternativa aos tradicionais *p*-valores freqüentistas, e não necessita, para sua aplicação, de nada além da distribuição *a posteriori* do(s) parâmetro(s), razão pela qual o teste é chamado *Full Bayesian* ("totalmente bayesiano"). O FBST apresenta também outras propriedades, elencadas por Faria Jr. (2006):

• fornece uma medida de significância da hipótese em teste, idealmente uma medida de probabilidade no espaço paramétrico do problema;
• tem uma interpretação inteiramente geométrica, invariante a parametrizações da hipótese ou ao sistema de coordenadas escolhido para o espaço paramétrico;
• sob condições apropriadas, fornece uma função de suporte contínuo e diferenciável nos parâmetros da hipótese e nas estatísticas da amostra;
• obedece ao princípio da verossimilhança;
• não utiliza artifícios *ad hoc* como a atribuição de probabilidades positivas a conjuntos e hipóteses de medida nula;
• é um procedimento exato, prescindindo de resultados ou aproximações assintóticas;
• é um teste consistente para hipóteses precisas;
• permite a incorporação de experiências, crenças ou opiniões pessoais, através de distribuições *a priori*.

Posteriormente, Madruga *et al* (2001) abordaram o FBST sob a ótica da Teoria da Decisão e comprovaram ser o FBST um legítimo teste de Bayes e explicitaram suas funções de perda (do tipo $L(d,\theta,x)$) específicas.

A construção desse teste, contudo, é intrinsecamente estática, uma vez que o cálculo da evidência só pode ser efetuado após a amostra ter sido inteiramente coletada e a distribuição *a posteriori* dos parâmetros, obtida. Em muitos casos, essa característica pode acarretar um custo amostral desnecessariamente alto, em decorrência da coleta de mais observações do que teria sido suficiente para a conclusão do teste.

Esse aspecto sugere forte e diretamente uma extensão do FBST ao contexto da Teoria da Decisão Seqüencial (DeGroot, 1970 e 2004). Tal extensão, na medida em que alia as propriedades supra listadas do FBST às vantagens abaixo relacionadas dos procedimentos

bayesianos seqüenciais (Klarmann, 1996), se mostra uma ferramenta extremamente promissora para a solução dos mais diversos problemas de inferência estatística:

• os procedimentos seqüenciais bayesianos são flexíveis, permitindo que inferências sejam realizadas a qualquer momento do processo amostral;

• por conseguinte, os procedimentos seqüenciais bayesianos permitem que a amostragem seja encerrada imediatamente, tão logo os dados evidenciem um diagnóstico (aceitação ou rejeição de H_0);

• diferentemente dos procedimentos seqüenciais clássicos, a abordagem bayesiana não viola o Princípio da Verossimilhança, já que a inferência realizada não guarda qualquer dependência ou subordinação a protocolos amostrais.

No capítulo 2, é apresentada a construção do FBST, conduzida tanto analítica quanto geometricamente, além de sua abordagem via Teoria da Decisão e da exibição de sua função de perda $L(d, \theta, x)$.

No capítulo 3, é feita uma revisão da Teoria de Decisão (baseada em DeGroot, 1970 *apud* Klarmann, 1996), desde o conceito de Problema de Decisão Puro até a teoria de Decisão Seqüencial.

O capítulo 4 aborda a aplicação ao FBST da teoria desenvolvida no capítulo 3, com enfoque especial nos problemas em que as distribuições (*priori/posteriori* e verossimilhança) sejam conjugadas. No capítulo 5 são tratadas as aplicações do FBST Seqüencial a problemas específicos, freqüentemente abordados na literatura

Um *software* para execução do FBST Seqüencial é desenvolvido em linguagem R (e seu código-fonte se encontra no Apêndice A) e o capítulo 6 reúne exemplos de aplicações desse *software* aos problemas abordados nos dois capítulos anteriores, além de observações sobre os resultados desses exemplos.

Por fim, no capítulo 7 são tecidos comentários e considerações finais acerca da teoria e do *software* desenvolvidos nesta tese.

Capítulo 2
FBST - Full Bayesian Significance Test

2.1. Construção do teste

Seja *X* uma variável aleatória que, quando observada, produz os dados *x* e considere-se um espaço $(\Xi, \boldsymbol{A}, \Theta, F, \xi)$, onde:

- Ξ é o espaço amostral, o conjunto dos possíveis valores de *x*;
- $\boldsymbol{A}$ é uma família de subconjuntos mensuráveis de Ξ;
- Θ é o espaço paramétrico;
- *F* é uma classe de medidas de probabilidade em $\boldsymbol{A}$, parametrizadas em Θ.
- ξ é uma medida de probabilidade *a priori* em (uma classe de subconjuntos mensuráveis de) Θ.

Considere-se também uma hipótese nula $H_0 : \theta \in \Theta_0$, onde $\Theta_0 \subset \Theta$.

O FBST - *Full Bayesian Significance Test* é, então, construído da seguinte forma (Pereira e Stern, 1999): considerando que a variável aleatória *X* segue uma distribuição *f* com parâmetro (escalar ou vetorial) θ e notando-se a distribuição *a priori* de θ por $\xi(\theta)$, pode-se representar a verossimilhança gerada pelos dados *x* por $f(x \mid \theta)$ e a distribuição *a posteriori* de θ por $\xi(\theta \mid x)$.

Estabeleça-se, então, para qualquer valor $\varphi \in [0,1]$, o conjunto T_φ, definido como o subconjunto do espaço paramétrico Θ onde a densidade *a posteriori* $\xi(\theta \mid x)$ é maior que φ:

$$T_\varphi = \{\theta \in \Theta \mid \xi(\theta \mid x) > \varphi\}$$

Em seguida, para qualquer conjunto T_φ, defina-se sua credibilidade como sendo a sua probabilidade *a posteriori*:

$$K(T_\varphi) = \int_{T_\varphi} \xi(\theta \mid x) d\theta$$

Definam-se, agora:

- θ^* como o (ou um) argumento onde a densidade *a posteriori* atinge o valor máximo sob a hipótese nula:

$$\theta^* \in \arg\max_{\theta \in \Theta_0} \xi(\theta \mid x).$$

- ξ^* como o valor dessa densidade:

$$\xi^* = \xi(\theta^* \mid x)$$

- T^* como o "conjunto tangente" à hipótese nula:

$$T^* = \{\theta \in \Theta \mid \xi(\theta \mid x) > \xi^*\}$$

Então, a credibilidade do "conjunto tangente" à hipótese nula é definida por

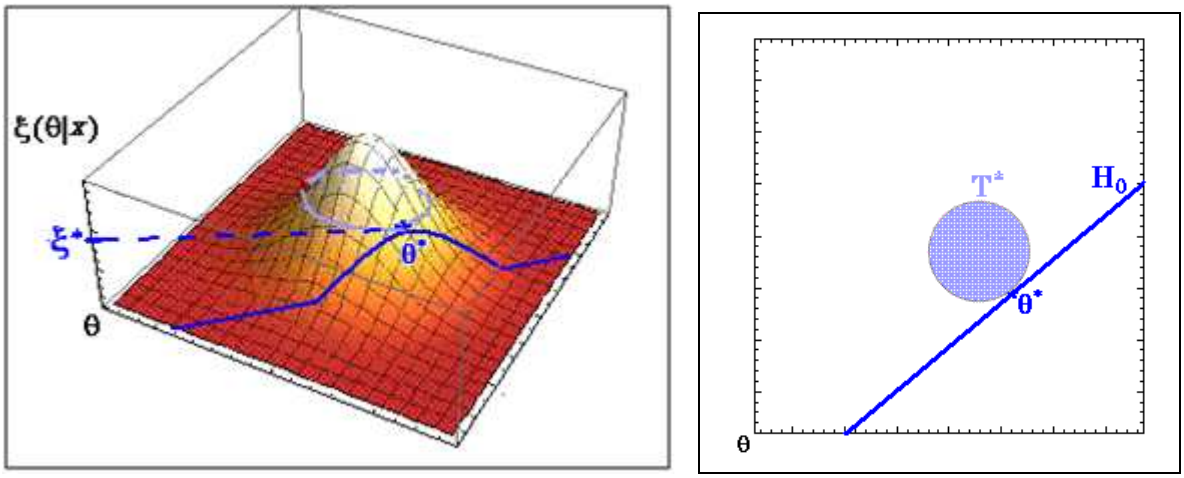
$$K^* = \xi(T^* \mid x) = \int_{T^*} \xi(\theta \mid x) d\theta$$

e, finalmente, a <u>evidência a favor</u> de Θ_0 (proporcionada pelos dados *x*) é definida por

$$\boxed{Ev(\Theta_0, x) = 1 - K^*}$$

Tomando como exemplo um espaço paramétrico bidimensional, a Figura 1 a seguir ilustra a interpretabilidade geométrica do FBST: sendo θ^* o ponto (indicado pelo asterisco) de máxima densidade *a posteriori* em Θ_0 (conjunto representada pela curva azul), o plano $\xi(\theta \mid x) = \xi^* = \xi(\theta^* \mid x)$ determina um "corte" em Θ, definido o conjunto tangente T^* (delimitado no gráfico (a) pela curva cinza-azulada e identificado no gráfico (b) pela região da mesma cor).

Então, T^* é o conjunto de todos os pontos de Θ que estejam graficamente "acima" de θ^* e sua credibilidade $K^* = \xi(T^* \mid x)$ é o volume, nessa região do espaço paramétrico, sob a curva de densidade *a posteriori*.



(a) Gráfico da densidade *a posteriori* (b) Plano paramétrico

Figura 1 - Visualização geométrica do FBST

Assim, se a medida $\xi(T^* \mid x)$ é "grande", isso significa que existem "muitos" pontos "mais prováveis" que θ^*. Logo, Θ_0 jaz em uma região de baixa probabilidade e, conseqüentemente, é baixa a evidência proporcionada pelos dados a favor da hipótese nula.

Analogamente, se a medida $\xi(T^* \mid x)$ é "pequena", isso significa que existem "poucos" pontos "mais prováveis" que θ^*. Logo, Θ_0 jaz em uma região de alta probabilidade e, conseqüentemente, é alta a evidência proporcionada pelos dados a favor da hipótese nula.

Essa construção é suficientemente geral, mas foi concebida visando, mais especificamente, testar hipóteses nulas precisas, ou seja, aquelas que correspondem a um subconjunto de dimensão inferior à do espaço paramétrico, ou seja, em que $\dim\Theta_0 < \dim\Theta$.

2.2. Funções de perda

Considerando-se um espaço de decisões $D = \{aceitar\ H_0\ ,\ rejeitar\ H_0\}$, Madruga *et al* (2001) mostram que o FBST é um teste de Bayes com funções de perda definidas em $D \times \Theta$ (vide capítulo 3) como segue:

$$\begin{cases} L(rejeitar\ H_0, \theta) = a[1 - \mathbf{1}\{\theta \in T^*\}] \\ L(aceitar\ H_0, \theta) = b + c\,\mathbf{1}\{\theta \in T^*\} \end{cases}$$, onde *a*, *b* e *c* são constantes positivas.

Os riscos (perdas esperadas) para cada decisão possível são, então, dados por:

$$\begin{cases} E_\xi[L(rejeitar\ H_0, \theta) \mid x] = a\,Ev(\Theta_0, x) \\ E_\xi[L(aceitar\ H_0, \theta) \mid x] = b + c\,(1 - Ev(\Theta_0, x)) \end{cases}$$

Conseqüentemente, o FBST equivale a um procedimento de minimização do risco esperado, pois a hipótese nula será aceita se e somente se

$$E_\xi[L(aceitar\ H_0, \theta) \mid x] < E_\xi[L(rejeitar\ H_0, \theta) \mid x],$$

o que equivale a aceitar H_0 se e somente se $\boxed{Ev(\Theta_0, x) > \frac{b+c}{a+c}}$.

Essa condição corrobora a interpretação inicial de que, quanto maior for o valor de $Ev(\Theta_0, x)$, maior será a propensão à aceitação da hipótese nula.

É importante notar que a valores diferentes de *x* correspondem conjuntos tangentes T^* potencialmente diferentes e a conjuntos tangentes diferentes correspondem funções de perda potencialmente distintas. Logo, as funções de perda para o FBST dependem dos valores amostrados *x*, o que pode, ao menos à primeira vista, colocar em dúvida a "bayesianidade" do teste.

Contudo, funções de perda que dependam dos valores amostrados podem ser capazes de incorporar alguns aspectos psicológicos com respeito a ordens individuais de preferência (Madruga *et al*, 2001), além de já haver registro de tais funções na literatura (como por exemplo em Bernardo e Smith, 1994).

Capítulo 3

Tópicos de Teoria de Decisão

Neste capítulo são apresentados e revisados elementos da Teoria de Decisão, de forma a solidificar o arcabouço sobre o qual se desenvolverá, no próximo capítulo, o FBST Seqüencial. Este capítulo é baseado na abordagem realizada por Klarmann (1996), por sua vez fundamentada predominantemente em DeGroot (1970).

3.1. Problema de decisão puro

O primeiro e mais básico tópico a ser abordado é o problema de decisão puro. Trata-se do caso mais simples possível, em que não há dados amostrais e a decisão deve ser tomada com base unicamente em uma distribuição *a priori*.

Notando-se por D o espaço de decisões, por Θ o espaço de estados da natureza (por exemplo, dos possíveis valores que um parâmetro θ pode assumir), por $L(d,\theta)$ a função de perda associada a cada par do cartesiano $D\times\Theta$ e por $\xi(\theta)$ a distribuição *a priori* do parâmetro/estado da natureza θ, definem-se:

- A Perda Esperada (ou Risco) de uma decisão *d* contra a *priori* ξ como:

$$\rho(\xi,d)=E[L(\theta,d)]=\begin{cases}\int_{\Theta}L(\theta,d)\xi(\theta)d\theta & \text{se } \Theta \text{ é uma v.a. contínua}\\ \sum_{i}L(\theta_i,d)\xi(\theta) & \text{se } \Theta \text{ é uma v.a. discreta}\end{cases} ;$$

- o Risco de Bayes (ou Risco de Decisão Ótima) contra a *priori* ξ como:

$$\rho^{*}(\xi)=\inf_{d\in D}\rho(\xi,d) \ ;$$

- a Decisão de Bayes (ou Decisão Ótima) contra a *priori* ξ, notada por d^{*}, como sendo qualquer decisão cujo risco é igual ao Risco de Bayes, ou seja, qualquer decisão que minimize o risco:

$$\rho(\xi,d^{*})=\rho^{*}(\xi) \ .$$

A solução do problema de decisão puro é, portanto, a simples busca da decisão que minimize a Perda Esperada contra a *priori* escolhida.

3.2. Problema de decisão estatística com tamanho de amostra fixo

O problema de decisão estatística difere do problema de decisão puro pela existência de uma amostra de uma variável aleatória *X*, relacionada ao parâmetro/estado da natureza em estudo. O caso mais simples desse problema é aquele em que o tamanho de amostra *n* é previamente fixado.

Assim, a um problema de decisão estatística, acrescentam-se aos elementos do problema de decisão puro os valores observados $x = (x_1, x_2, \ldots, x_n)$ da variável aleatória *X* e define-se, também, a Regra de Decisão $\delta(x)$ como a função que associa a cada possível vetor observado *x* a decisão $d \in D$ a ser tomada. Nota-se por Δ o espaço de todas as regras de decisão possíveis.

Por fim, representa-se por *C* o custo de amostragem (aqui suposto constante) para cada observação individual. Usualmente, costuma-se representar na literatura esse custo pela letra *c* minúscula. Nesta tese, porém, será utilizada a letra *C* maiúscula, a fim de evitar confusões com o parâmetro *c* (minúsculo) relacionado à função de perda do FBST (Seções 2.2 e 4.3). Assim, o Custo Amostral da observação do vetor $x = (x_1, x_2, \ldots, x_n)$ será, naturalmente, igual a *nC*.

Nesse contexto, a Função de Perda Total, notada por L_t é dada por:

$$L_t(\delta, \theta, C) = L(\delta(x), \theta) + nC$$

e pode ser decomposta em duas parcelas, com significados próprios:

a) $L(\delta(x), \theta)$, a parcela que depende de θ e de δ, referida como Perda de Decisão e equivalente à função de perda do correspondente problema de decisão puro;
b) *nC*, o Custo Amostral, parcela que depende somente do tamanho da amostra.

Um problema de decisão estatística é, então, definido pelos espaços Δ e Θ supra definidos, pelo espaço amostral Ξ (definido na Seção 2.1), pela função de Perda de Decisão *L* e pela distribuição $h(\theta, x) = f(x \mid \theta)\xi(\theta)$, a *priori* conjunta para θ e *x*.

Um dado problema de decisão estatística com tamanho fixo de amostra pode ser analisado de duas formas possíveis, as quais serão abordadas nas subseções a seguir.

3.2.1. Análise Pós-Experimental

Essa abordagem, analiticamente a mais simples, consiste em calcular os riscos e tomar a decisão após a coleta da amostra *x*. Trata-se, em outras palavras, de simplesmente abordar o problema correspondente de decisão puro, considerando como *priori* para θ a distribuição *a posteriori* $\xi(\theta \mid x)$, também notada como $\xi_x(\theta)$. Dessa forma, nessa abordagem tem-se:

- Perda Esperada (Risco) de uma decisão $\delta(x)$ contra a *posteriori* $\xi(\theta \mid x)$:

$$\rho[\xi_x, \delta(x)] = E[L(\theta, \delta(x))] = \int_\Theta L[\theta, \delta(x)]\xi_x(\theta)d\theta \quad .$$

Cabe observar, a propósito, que a quase totalidade das distribuições *a posteriori* usualmente encontradas são contínuas. Por isso, doravante se considerará tacitamente ξ e ξ_x contínuas e não será abordado o caso análogo em que tais distribuições sejam discretas.

- Risco de Bayes (Risco de Decisão Ótima ou Risco de Bayes *a Posteriori*) contra a *posteriori* ξ_x:

$$\rho^*(\xi_x) = \inf_{\delta(x)\in D} \rho[\xi_x, \delta(x)] \ ;$$

- Decisão de Bayes (Decisão Ótima) contra a *posteriori* ξ_x, notada por $\delta^*(x)$: qualquer decisão cujo risco seja igual a $\rho^*(\xi_x)$.

Por fim, define-se o Risco Total de Decisão de Bayes (ou Risco Total de Bayes *a Posteriori*), notado por $R^*(\xi_x)$, como:

$$R^*(\xi_x) = \inf_{\delta(x)\in D} \left(E[L[\theta, \delta(x)]] + nC\right) = \rho^*(\xi_x) + nC \ .$$

Assim, a solução do problema de decisão é, nesse caso, simplesmente a determinação da decisão que minimize a Perda Esperada contra a *posteriori* obtida após a amostragem.

3.2.2. Análise Pré-Experimental

Essa abordagem (também conhecida como Análise Pré-Posterior) difere da anterior por consistir em calcular os riscos para cada decisão possível antes da coleta da amostra *x*. Tal abordagem permite não somente identificar a decisão ótima segundo cada observação amostral possível, mas também avaliar se se deve ou não realizar tal amostragem. Assim, nessa abordagem tem-se:

- Perda Esperada (Risco) de uma decisão δ contra a *priori* conjunta $h(\theta, x) = f(x \mid \theta)\xi(\theta)$:

$$\rho(h, \delta) = E[L(\theta, \delta(X))] = \int_\Theta \int_\Xi L[\theta, \delta(x)]\xi(\theta) f(x \mid \theta) dx d\theta \ ;$$

- Risco de Bayes (Risco de Decisão Ótima ou Risco de Bayes *a Priori*) contra a *priori* conjunta *h*, notado por $\rho^*(h)$:

$$\rho^*(h) = \inf_{\delta\in\Delta} \rho(h, \delta) \ ;$$

Define-se, então, uma Regra de Decisão de Bayes contra a *priori* conjunta $h(\theta, x) = f(x \mid \theta)\xi(\theta)$, notada por δ^*, como sendo qualquer regra de decisão cujo risco $\rho(h, \delta^*)$ seja igual a $\rho^*(h)$.

Por fim, o Risco Total de Bayes *a Priori*, notado por $R^*(h)$, é dado por

$$R^*(h) = \rho^*(h) + nC \ .$$

Uma maneira de solucionar esse problema de decisão é por meio da minimização direta de $R^*(h)$ em relação a δ. Essa minimização, conhecida como Forma Normal para procura de uma regra de decisão de Bayes, freqüentemente se torna computacionalmente complexa. O resultado a seguir (Klarmann, 1996) fornece um outro método, conhecido como Forma Extensiva:

$$R^*(h) = nC + E[\rho(\xi_X, \delta(X))] \ .$$

Isso significa que pode-se obter a regra ótima δ^* que soluciona o problema de decisão estatística simplesmente considerando, para cada amostra possível *x*, o problema puro correspondente, ou seja, tomando como distribuição *a priori* para θ a respectiva *posteriori* $\xi(\theta \mid x)$.

Essa é uma conclusão bastante intuitiva, uma vez que, para qualquer amostra particular x_0, não há razão para que a regra ótima $\delta^*(x_0)$ obtida por meio da Análise Pós-Experimental difira da regra ótima δ^* encontrada via Análise Pré-Experimental e aplicada ao valor/vetor x_0.

3.3. Problema de decisão estatística com amostragem seqüencial

As soluções apresentadas ao problema de decisão estatística abordado na seção anterior dependem direta e fundamentalmente de o tamanho da amostra ser fixo e previamente estabelecido. Em situações concretas, porém, muitas vezes se tem acesso a informações parciais da amostra (ou mesmo o processo amostral pode ser interrompido antes do momento previsto, por motivos alheios ao protocolo estabelecido) e uma abordagem comprometida com um tamanho de amostra previamente fixado pode, nesse caso, acarretar desperdício de informações coletadas e/ou um custo de amostragem desnecessariamente alto.

Por essa razão, emprega-se em situações como essas uma abordagem seqüencial, em que, após cada observação ser coletada, a informação (a *posteriori*) sobre θ é atualizada e nova análise é realizada a fim de se decidir entre o término da amostragem ou a tomada de mais uma observação.

3.3.1. Definições

Nesse contexto:

- $\mathbf{X} = (X_1, X_2, X_3, \ldots)$ representa um processo estocástico,
- $\mathbf{X}_n = (X_1, X_2, \ldots, X_n)$ representa uma amostra seqüencial de tamanho *n*
- e as variáveis aleatórias $X_1, X_2, \ldots$ são supostas condicionalmente (dado θ) independentes e identicamente distribuídas com distribuição comum $f_{X_1}(\cdot \mid \theta)$.

Além disso, um problema de decisão estatística com amostragem seqüencial possui duas componentes básicas:

a) Regra de Parada: definida como uma seqüencia de funções $\tau = (\tau_0, \tau_1, \tau_2, \ldots)$ que satisfazem, para todo $n = 0,1,2,\ldots$, as seguintes propriedades:

i) $\tau_n \in \{0,1\}$ (onde $\tau_n = 1$ equivale a "pare a amostragem" e $\tau_n = 0$ equivale a "continue a amostragem) ;

ii) $\tau_n = \tau(X_1, X_2, \ldots, X_n)$;

iii) $\tau_n = 1 \Rightarrow \tau_{n+1} = 1$.

Trata-se, portanto, de uma função que associa a cada amostra $(X_1, X_2, \ldots, X_n)$ uma determinação entre continuar ou parar a amostragem. As propriedades (i) e (ii) caracterizam matematicamente a função τ_n, enquanto a propriedade (iii) nada mais é do que a tradução da noção intuitiva de que, se a determinação associada a uma amostra $(X_1, X_2, \ldots, X_n)$ é de parar a amostragem, então a determinação associada a $(X_1, X_2, \ldots, X_n, X_{n+1})$ também será de parar a amostragem, qualquer que seja o valor de X_{n+1}. Em outras palavras, a decisão de encerramento da amostragem é sempre definitiva e irreversível.

A partir do conceito de Regra de Parada, pode-se definir as Regiões de Parada $B_1, B_2, \ldots$ como:

$$B_n = \left\{(x_1, \ldots, x_n) \in \Xi^n : \tau_n(x_1, \ldots, x_n) = 1\right\}.$$

Ou seja: B_n é o conjunto de todas as amostras possíveis para as quais a determinação de parada da amostragem ocorre, no máximo, imediatamente após a *n*-ésima observação. Naturalmente, se $(x_1, \ldots, x_n) \in B_n$, então vetores do tipo $(x_1, \ldots, x_n, x_{n+1})$, $(x_1, \ldots, x_n, x_{n+1}, x_{n+2})$ etc. não precisarão ser levados em consideração, uma vez que a $(n+1)$-ésima e as subseqüentes observações não serão coletadas.

b) Regra de Decisão Seqüencial: definida como uma seqüencia de funções $\delta = (\delta_0, \delta_1, \delta_2, \ldots)$, onde $\delta_0 \in D$ e $\delta_n = \delta(X_1, X_2, \ldots, X_n) \in D$ para todo $n = 1,2, \ldots$

A regra $\delta_n = \delta(X_1, X_2, \ldots, X_n)$ estabelece, então, qual será, para cada amostra possível $(x_1, \ldots, x_n) \in B_n$, a decisão terminal $d \in D$ a ser tomada ao término da amostragem.

Por fim, definem-se:

- O tamanho de amostra associado a uma regra de parada τ como a variável aleatória

$$N_\tau = \min_{n \geq 0}\{n : \tau_n = 1\} .$$

- Um procedimento de decisão seqüencial γ como o par de regras (de parada e de decisão) (τ, δ).

- A Função de Perda Total de um problema de decisão seqüencial, dada por:

$$L_t(\delta, \tau, \theta, C, n) = L[\delta_{N_\tau}(\theta, X_1, \ldots, X_{N_\tau})] + CN_\tau .$$

Um problema de decisão estatística com amostragem seqüencial é, então, definido por Γ (o espaço de todos os procedimentos de decisão seqüenciais γ possíveis), pelos espaços Θ e Ξ anteriormente definidos, pela função de Perda Total L_t e pela distribuição a *priori* conjunta para θ e $\mathbf{X} = (X_1, X_2, \ldots, X_{N_\tau})$, dada por $g(\theta, \mathbf{X}) = \xi(\theta) \prod_{i=1}^{N_\tau} f(X_i \mid \theta)$.

3.3.2. Procedimentos de decisão seqüenciais de Bayes

Supondo-se $\tau_0 = 0$ (ou seja, que pelo menos uma observação deverá ser tomada), a Perda Esperada Total (ou Risco Total) de um procedimento de decisão seqüencial γ contra a *priori* $g(\theta, \mathbf{X})$ é dado por:

$$R(g, \gamma) = E[L(\theta, \delta_{N_\tau}(X_1, \ldots, X_{N_\tau})) + CN_\tau] \ .$$

Analogamente aos problemas anteriores, pode-se definir:

- Risco Total de Bayes *a Priori* de um problema seqüencial com *priori* $g(\theta, \mathbf{X})$:

$$R^*(g) = \inf_{\gamma \in \Gamma} R(g, \gamma) \ .$$

- Procedimento de Decisão Seqüencial de Bayes contra a *priori* $g(\theta, \mathbf{X})$, notado por γ^*: qualquer procedimento cujo risco $R(g, \gamma^*)$ seja igual a $R^*(g)$.

Assim, a solução de um problema de decisão seqüencial é a determinação de um procedimento γ^* que minimize o risco $R(g, \gamma)$. A busca direta de tal procedimento ótimo tende a ser um trabalho bastante complexo, dada a infinidade de procedimentos γ possíveis a serem considerados.

Klarmann (1996) mostra que se $\gamma^* = (\delta^+, \tau^+) = [(\delta_0^+, \tau_0^+), (\delta_1^+, \tau_1^+), \ldots]$ é um procedimento de decisão seqüencial de Bayes contra a *priori* $g(\theta, \mathbf{X})$, então para todo $n = 1,2...$ e para quase todo $(x_1, \ldots, x_n) \in \Xi^n$, a decisão $\delta_n^+(x_1, \ldots . x_n)$ é uma decisão de Bayes contra a *posteriori* $\xi_n(\theta \mid x_1, \ldots, x_n)$.

Esse resultado facilita sensivelmente a busca do procedimento ótimo γ^* pois, em outras palavras, para todo $n = 1,2...$, a regra de decisão δ_n do procedimento seqüencial de Bayes é exatamente a regra de decisão de Bayes δ_n^* Isso reduz a tarefa (minimização de $R^*(g)$) à determinação da regra de parada ótima pois, uma vez que essa regra determine o término da amostragem após *n* observações, a decisão a ser tomada será exatamente a decisão de Bayes contra a *posteriori* ξ_n.

3.4. Procedimentos de decisão seqüenciais truncados

3.4.1. Notações

Nesta seção serão apresentadas soluções ao problema da determinação do (ou de um) Procedimento de Decisão Seqüencial de Bayes. Para maior simplicidade no desenvolvimento dessas soluções, doravante será utilizada a seguinte notação:

- Risco de Bayes contra a *priori* ξ_0 para θ (como no problema puro correspondente):

$$\rho_0(\xi_0) = \inf_{d \in D} \int_\Theta L(\theta, d)\xi_0(\theta)d\theta \qquad (1)$$

- Risco de Bayes a *posteriori* contra ξ_n (como no problema correspondente de decisão estatística com tamanho de amostra fixo):

$$\rho_0(\xi_n) = \inf_{d \in D} \int_\Theta L(\theta, d)\xi_n(\theta)d\theta \qquad (2)$$

- Risco Total de Bayes a *Posteriori* contra ξ_n:

$$R_0(\xi_n) = \begin{cases} \rho_0(\xi_0) & \text{para } n = 0 \\ \rho_0(\xi_n) + nC & \text{para } n = 1,2,\ldots \end{cases}$$

- Risco Total de Bayes a *Priori* contra $g(\theta, \mathbf{X})$:

$$R_0(g) = \begin{cases} \rho_0(\xi_0) & \text{para } n = 0 \\ E[\rho_0(\xi_{N_\tau}) + CN_\tau] & \text{para } n = 1,2,\ldots \end{cases}$$

3.4.2. Método da Indução **Backward**

A forma mais direta de construção do Procedimento de Decisão Seqüencial de Bayes seria uma análise que comparasse, inicialmente, o risco da tomada imediata de decisão com o risco da coleta de uma observação (X_1). Ocorre, porém, que a avaliação do risco da observação de X_1 exige considerar tanto a possibilidade de uma tomada imediata de decisão após essa observação, quanto a possibilidade de se tomar mais uma observação (X_2). Da mesma forma, o estudo do risco da observação de (X_1, X_2) exige que se leve em conta tanto a possibilidade de uma tomada de decisão imediatamente após essas duas observações, quanto a possibilidade de se coletar mais uma observação (X_3). Assim sucessivamente, essa abordagem poderia se estender indefinidamente, tornando sua aplicação virtualmente impraticável.

Uma solução alternativa se baseia no truncamento do processo amostral, limitando a quantidade de observações coletadas a um máximo de *M*. Dá-se a tais procedimentos o nome de Procedimentos de Decisão Seqüenciais Truncados (ou *M*-Truncados, quando se referindo

especificamente àqueles limitados a *M* observações). A construção de um Procedimento de Decisão Seqüencial *M*-Truncado de Bayes se dá por meio da técnica de Indução *Backward*, a qual consiste em partir do risco da tomada de decisão após a coleta de todas as *M* observações e ir retrocedendo etapa por etapa do processo amostral até chegar (no máximo) ao risco da tomada imediata (sem a coleta de qualquer observação) de decisão.

Chamando de continuação ótima o prosseguimento adotado de acordo com o Princípio de Bayes (princípio da minimização do risco), os passos abaixo descrevem detalhadamente a técnica de Indução *Backward*:

• **Passo 1:** consiste em considerar que $X_1, X_2, \ldots, X_M$ tenham sido observados. Nesse caso, a continuação ótima é encerrar a amostragem e tomar a decisão terminal $d \in D$ escolhida com base nos valores $(x_1, x_2, \ldots, x_M)$.

O Risco de Bayes dessa continuação ótima é, conforme a equação (2), $\rho_0(\xi_M)$.

• **Passo 2:** agora, supõe-se que $X_1, X_2, \ldots, X_{M-1}$ tenham sido observados e o objetivo é comparar entre o encerramento imediato da amostragem e a tomada da *M*-ésima observação.

Caso a amostragem seja encerrada nesse momento, a decisão terminal será tomada com base nos valores $(x_1, x_2, \ldots, x_{M-1})$ e seu Risco de Bayes será, conforme a equação (2), igual a $\rho_0(\xi_{M-1})$.

Caso seja tomada mais uma (a *M*-ésima) observação, a distribuição utilizada para a tomada de decisão será a *posteriori* ξ_{M-1} atualizada pelo valor observado x_M, o que será doravante notado por $\xi_{M-1}(x_M)$. Assim, para cada possível valor de x_M, o Risco de Bayes será, conforme a equação (2), igual a $\rho_0(\xi_{M-1}(x_M))$ e o Risco Total será $\rho_0(\xi_{M-1}(x_M)) + C$. Logo, o risco do prosseguimento da amostragem é $E[\rho_0(\xi_{M-1}(X_M))] + C$.

Assim, a regra de parada do procedimento de decisão seqüencial *M*-truncado de Bayes estabelece que, caso a observação X_{M-1} tenha sido coletada, então:

- Se $\rho_0(\xi_{M-1}) \leq E[\rho_0(\xi_{M-1}(X_M))] + C$, então a continuação ótima é parar a amostragem e tomar a decisão terminal em *D* com base nas observações $(x_1, x_2, \ldots, x_{M-1})$;

- Se $\rho_0(\xi_{M-1}) > E[\rho_0(\xi_{M-1}(X_M))] + C$, então a continuação ótima é observar X_M e tomar a decisão terminal em *D* com base nas observações $(x_1, x_2, \ldots, x_{M-1}, x_M)$.

Por fim, o risco da continuação ótima dados $x_1, x_2, \ldots, x_{M-1}$ é, então, igual a:

$$\rho_1(\xi_{M-1}) = mín\{\rho_0(\xi_{M-1}) ; E[\rho_0(\xi_{M-1}(X_M))] + C\} \ .$$

• **Passo 3:** analogamente ao passo anterior, supõe-se, agora, que $X_1, X_2, \ldots, X_{M-2}$ tenham sido observados e o objetivo é comparar entre o encerramento imediato da amostragem e a tomada da $(M-1)$-ésima observação.

Semelhantemente ao passo anterior, caso a amostragem seja encerrada nesse momento, a decisão terminal será tomada com base nos valores $(x_1, x_2, \ldots, x_{M-2})$ e seu Risco de Bayes será igual a $\rho_0(\xi_{M-2})$.

Também semelhantemente ao passo anterior, caso seja tomada a $(M-1)$-ésima observação, a distribuição utilizada para a tomada de decisão será a *posteriori* $\xi_{M-2}(x_{M-1})$. Agora, porém, o Risco de Bayes para cada possível observação x_{M-1} será $\rho_1(\xi_{M-2}(x_{M-1}))$ e, por fim, o risco do prosseguimento da amostragem é dado por $E[\rho_1(\xi_{M-2}(X_{M-1}))]+C$.

Assim, a regra de parada do procedimento de decisão seqüencial *M*-truncado de Bayes determina que, caso a observação X_{M-2} tenha sido coletada, então:

- Se $\rho_0(\xi_{M-2}) \leq E[\rho_1(\xi_{M-2}(X_{M-1}))]+C$, então a continuação ótima é parar a amostragem e tomar a decisão terminal em *D* com base nas observações $(x_1, x_2, \ldots, x_{M-2})$;

- Se $\rho_0(\xi_{M-2}) > E[\rho_1(\xi_{M-2}(X_{M-1}))]+C$, então a continuação ótima é observar X_{M-1} e, depois, seguir o estabelecido no Passo 2.

Logo, o risco da continuação ótima dados $x_1, x_2, \ldots, x_{M-2}$ é dado por:

$$\rho_2(\xi_{M-2}) = mín\{\rho_0(\xi_{M-2}) ; E[\rho_1(\xi_{M-2}(X_{M-1}))]+C\} \ .$$

- **Passos Seguintes:** repete-se o mesmo processo, sempre comparando-se o risco de interrupção imediata da amostragem com o da tomada de mais uma observação.

Caso a amostragem seja encerrada após terem sido observados $X_1, X_2, \ldots, X_{M-j}$, por exemplo, a decisão terminal será tomada com base nos valores $(x_1, x_2, \ldots, x_{M-j})$ e seu Risco de Bayes será igual a $\rho_0(\xi_{M-j})$.

No mesmo exemplo, caso seja tomada a $(M-j+1)$-ésima observação, o risco do prosseguimento da amostragem será $E[\rho_{j-1}(\xi_{M-j}(X_{M-j+1}))]+C$.

Logo, a regra de parada do procedimento de decisão seqüencial *M*-truncado de Bayes estabelece que, caso a observação X_{M-j} tenha sido coletada, então:

- Se $\rho_0(\xi_{M-j}) \leq E[\rho_{j-1}(\xi_{M-j}(X_{M-j+1}))]+C$, então a continuação ótima é parar a amostragem e tomar a decisão terminal com base nas observações $(x_1, x_2, \ldots, x_{M-j})$;

- Se $\rho_0(\xi_{M-j}) > E[\rho_{j-1}(\xi_{M-j}(X_{M-j+1}))]+C$, então a continuação ótima é observar X_{M-j+1} e, depois, seguir o estabelecido no "*j*-ésimo passo".

Aqui, o risco da continuação ótima dados $x_1, x_2, \ldots, x_{M-j}$ é dado por:

$$\rho_j(\xi_{M-j}) = mín\{\rho_0(\xi_{M-j}) ; E[\rho_{j-1}(\xi_{M-j}(X_{M-j+1}))]+C\} \ .$$

• **Passo Final:** o último passo consiste em comparar o risco da tomada de uma decisão imediata (sem observações) com o risco do início da amostragem (com a observação de X_1).

Nessa etapa, caso a decisão seja tomada imediatamente, sem observações, a decisão terminal será tomada com base na *priori* ξ_0 e seu Risco de Bayes será, conforme a equação (1), igual a $\rho_0(\xi_0)$.

Caso seja tomada a primeira observação, o risco do prosseguimento da amostragem será, analogamente aos passos anteriores, $E[\rho_{M-1}(\xi_0(X_1))]+C$.

Assim, a regra de parada do procedimento de decisão seqüencial *M*-truncado de Bayes determina que:

- Se $\rho_0(\xi_0) \leq E[\rho_{M-1}(\xi_0(X_1))]+C$, então a continuação ótima é tomar a decisão imediatamente, sem a coleta de observações;
- Se $\rho_0(\xi_0) > E[\rho_{M-1}(\xi_0(X_1))]+C$, então a continuação ótima é observar X_1 e, depois, seguir o estabelecido no "*M*-ésimo passo".

Por fim, o risco da continuação ótima nesse caso será dado por:

$$\rho_M(\xi_0) = mín\{\rho_0(\xi_0)\,;\, E[\rho_{M-1}(\xi_0(X_1))]+C\} \ .$$

3.4.3. O procedimento de decisão seqüencial M*-truncado de Bayes*

Semelhantemente aos conceitos anteriormente estabelecidos para os procedimentos de decisão seqüenciais em geral, definem-se, especificamente para os procedimentos seqüenciais *M*-truncados (notados por γ^M):

- Γ^M como o espaço de todos os procedimentos seqüenciais *M*-truncados;

- o Risco Total de Bayes *a Priori* de um problema seqüencial *M*-truncado com *priori* $g_M(\theta, \mathbf{X}_M)$, como:

$$R^*(g_M) = \inf_{\gamma^M \in \Gamma^M} R(g_M, \gamma^M) \ \ ;$$

- o Procedimento de Decisão Seqüencial *M*-truncado de Bayes contra a *priori* $g_M(\theta, \mathbf{X}_M)$, notado por γ^{M*}, como qualquer procedimento cujo risco $R(g_M, \gamma^{M*})$ seja igual a $R^*(g_M)$.

Klarmann (1996) apresenta, então, a relação a seguir (DeGroot, 1970), válida para qualquer etapa $n < M$ de um procedimento de decisão seqüencial *M*-truncado de Bayes:

$$\rho_{M-n}(\xi_n) = R^*(g_{M-n}) \text{, onde:}$$

- $\rho_{M-n}(\xi_n)$ é o risco da continuação ótima na etapa *n* (isto é, dado terem sido observados $x_1,\ldots,x_n$) de um problema de decisão seqüencial *M*-truncado de Bayes com *priori* $g_M(\theta,\mathbf{X}_M)$;

- $R^*(g_{M-n})$ é o Risco Total de Bayes a *Priori* de um problema de decisão seqüencial $(M-n)$-truncado de Bayes com *prioris* $\xi_n(\theta)$ e $g_{M-n}(\theta,\mathbf{X}_{M-n})$.

Isso equivale a dizer que, se num problema seqüencial *M*-truncado, já foram tomadas *n* observações, então essa situação é rigorosamente equivalente a um problema seqüencial $(M-n)$-truncado com amostragem a ser iniciada (ou não) nesse exato momento, com *priori* $\xi_n(\theta)$.

Assim, com base na técnica de Indução *Backward*, no resultado acima e no resultado apresentado ao final da Seção 3.3.2, pode-se estabelecer que o Procedimento de Decisão Seqüencial *M*-truncado de Bayes determina que:

1 - Na etapa inicial do processo:

- Se $\rho_0(\xi_0)=\rho_M(\xi_0)$, então a continuação ótima é não iniciar a amostragem e escolher uma decisão terminal de Bayes d^* em *D* sem que observação alguma seja tomada;

- Se $\rho_0(\xi_0)>\rho_M(\xi_0)$, então X_1 deve ser observado.

2 - Nas etapas *j* $(j=1,\ldots,M-1)$ subseqüentes, supondo que $(x_1,\ldots,x_j)$ tenham sido observados:

- Se $\rho_0(\xi_j)=\rho_{M-j}(\xi_j)$, então a continuação ótima é terminar a amostragem nessa etapa *j* e escolher em *D* uma decisão terminal de Bayes contra ξ_j;

- Se $\rho_0(\xi_j)>\rho_{M-j}(\xi_j)$, então X_{j+1} deve ser observado.

3 - Se a amostragem prosseguir até a etapa *M*:

Então a continuação ótima é interromper a amostragem nessa etapa (após a observação de X_M e qualquer que seja o valor observado) e escolher em *D* uma decisão terminal de Bayes contra ξ_M.

3.4.4. Tamanho ótimo de amostra

Tendo sido determinado o Procedimento Seqüencial *M*-Truncado de Bayes γ_M^*, resta agora encontrar o valor de *M* que melhor aproxima tal procedimento do Procedimento Seqüencial de Bayes γ^*.

Klarmann (1996) elenca as condições a seguir, que podem auxiliar na determinação desse valor de *M* (em ambas, *C* é o custo por observação amostral):

Condição 1: Se o risco de Bayes *a posteriori* na etapa *k* satisfaz a relação

$$\rho_0(\xi_k) < C$$

para todo $(x_1,\ldots,x_k) \in \Xi^k$ e se *M* é o menor valor de *k* para o qual essa relação seja verdadeira, então o procedimento seqüencial ótimo é o procedimento *M*-truncado ótimo.

Essa condição, extremamente intuitiva, equivale a dizer, em linhas gerais, que, se para qualquer vetor observável $(x_1,\ldots,x_k)$ o risco da tomada imediata de decisão é menor que o custo da coleta da (*k*+1)-ésima observação, então a continuação ótima sempre será encerrar a amostragem após a *k*-ésima observação. E, particularmente, se *M* é o menor valor de *k* para qual esse fato se verifica, então o procedimento seqüencial ótimo é exatamente o procedimento *M*-truncado ótimo.

Condição 2: Se, para todo $j \geq k$ e para todo $(x_1,\ldots,x_j) \in \Xi^j$, a relação

$$\rho_0(\xi_j) \leq E[\rho_0(\xi_j(X_{j+1}))] + C$$

for satisfeita e se *M* é o menor valor de *k* para o qual essa relação seja verdadeira, então o procedimento seqüencial ótimo é o procedimento *M*-truncado ótimo.

Essa condição e análoga à anterior, diferindo, basicamente, apenas na comparação do risco da tomada imediata de decisão (após a observação de $(x_1,\ldots,x_k)$) com o risco do prosseguimento da amostragem (em lugar do custo da tomada de mais uma observação). Novamente, se *M* é particularmente o menor valor de *k* para qual esse fato se verifica, então o procedimento seqüencial ótimo é exatamente o procedimento *M*-truncado ótimo.

Nem sempre, contudo, será possível encontrar um valor de *M* que obedeça a alguma dessas condições e seja pequeno o suficiente para poder ser computacionalmente tratável.

Capítulo 4
FBST Seqüencial

4.1. Notações e definições

Notando-se por $x_1, x_2, x_3, \ldots$ os valores amostrados de X e invocando a notação apresentada na Seção 2.1, obtém-se, após cada valor ser observado, uma seqüência de evidências dada por $Ev(\Theta_0, x_1)$, $Ev(\Theta_0, (x_1, x_2))$, $Ev(\Theta_0, (x_1, x_2, x_3)), \ldots$

Como esses valores são obtidos a partir das sucessivas distribuições *a posteriori* $\xi_1, \xi_2, \xi_3 \ldots$ e como essas *posterioris* são determinadas unicamente a partir de $\xi(\theta)$ e das observações $x_1, x_2, \ldots$, pode-se representar, sem qualquer ambigüidade ou perda de informação, essa seqüência de evidências a favor de Θ_0 por $Ev(\Theta_0, \xi_1), Ev(\Theta_0, \xi_2)$, $Ev(\Theta_0, \xi_3), \ldots$ Também se pode, por extensão de raciocínio, representar por $Ev(\Theta_0, \xi_0)$ a evidência calculada a partir da *priori*, isto é, sem que nenhuma observação seja amostrada.

É importante salientar, neste ponto, que as distribuições *a posteriori* $\xi_1(\theta), \xi_2(\theta), \ldots$ e suas respectivas evidências a favor da hipótese nula dependem diretamente dos valores observados de $x_1, x_2, \ldots$. Logo, naturalmente, cada valor que x_1 puder assumir corresponderá a uma *posteriori* $\xi_1(\theta)$ específica e, conseqüentemente, a um valor específico da evidência $Ev(\Theta_0, \xi_1)$. Assim, no restante desta tese, quando essa dependência da *posteriori* e da evidência ao(s) valor(es) amostrado(s) for relevante, esta será claramente explicitada em notações como $\xi_0(X_1)$ ou $Ev(\Theta_0, \xi_0(X_1))$.

4.2. Abordagem analítica

Como uma primeira abordagem, poder-se-ia arbitrar um número *M* suficientemente grande e aplicar ao FBST o procedimento seqüencial *M*-truncado de Bayes, apresentado na Subseção 3.4.3. Dificilmente, porém, o valor de *M* poderá ser diretamente arbitrado a partir de características particulares do problema e/ou de conhecimentos de especialistas na área de aplicação em questão.

Assim, a alternativa mais imediata seria buscar, analiticamente, valores de *M* que obedeçam a alguma das Condições 1 e 2, apresentadas na Subseção 3.4.4. Contudo, a forma recursiva com que os riscos $\rho_\cdot(\xi_\cdot)$ são obtidos já dificulta, por si só, os cálculos envolvidos. No caso particular do FBST, tal dificuldade é bastante acentuada pelo fato de as evidências serem integrais sobre regiões dependentes dos valores amostrados, o que torna extremamente inviável a verificação analítica daquelas condições.

Uma outra possibilidade (referida doravante como Condição 3) é formulada por DeGroot (2004) que demonstra que, se existe um número *K* tal que, para qualquer vetor observável $(x_1, \ldots, x_n) \in B_n$ a relação $\rho_0(\xi_n) < nK$ é obedecida, então

$$\boxed{\lim_{n\to\infty} \rho(\xi_0, \gamma_n^*) = \rho(\xi_0, \gamma^*)}\,.$$

Assim, um possível procedimento básico para problemas de decisão seqüencial em que o resultado acima possa ser aplicado, pode ser sumarizado pela seguinte formulação analítica:

a) Calcule o risco $\rho_0(\xi_0)$ da tomada da decisão ótima sem a coleta de novas observações.
b) Calcule o risco $E[\rho_0(\xi_0(X_1))]$ do prosseguimento da amostragem com a coleta de uma nova observação, seguida da tomada da decisão terminal ótima.
c) Se $E[\rho_0(\xi_0(X_1))]+C<\rho_o(\xi_0)$, colete uma nova observação. Caso contrário, encerre a amostragem.

Essa formulação corresponde a adotar o procedimento γ_1^* (isto é, o procedimento de decisão seqüencial "1-truncado" de Bayes) tantas vezes quantas sejam necessárias até que a decisão ótima seja aquela tomada sem a coleta de novas observações.

Aplicando-se essa abordagem à determinação do procedimento seqüencial de Bayes aplicado ao FBST, pode-se verificar que:

como $\rho_0(\xi_n)=\min\{a\,Ev(\Theta_0,\xi_n);b+c(1-Ev(\Theta_0,\xi_n))\}$ (Seção 2.2)

e como, pela forma como o FBST é construído, $Ev(\Theta_0,\xi_n)\le 1$ (Seção 2.1),

então $\rho_0(\xi_n)\le\min\{a;b+c\}<\min\{a;b+c\}+\varepsilon<(\min\{a;b+c\}+\varepsilon)\cdot n$ ■

Logo, o FBST satisfaz a Condição 3 para $K=\min\{a;b+c\}+\varepsilon,\forall\varepsilon>0$ e o respectivo resultado pode ser aplicado. Conseqüentemente, pode-se também aplicar ao FBST o procedimento básico descrito mais acima, para o qual:

- $\rho_0(\xi_0)=\min\{a\,Ev(\Theta_0,\xi_0);b+c(1-Ev(\Theta_0,\xi_0))\}$
- $E[\rho_0(\xi_0(X_1))]=E[\min\{a\,Ev(\Theta_0,\xi_0(X_1));b+c(1-Ev(\Theta_0,\xi_0(X_1)))\}]$

Todavia, a expressão de $E[\rho_0(\xi_0(X_1))]$ depende da avaliação, para cada possível valor amostrado x_1, do valor da evidência para a respectiva *posteriori* $\xi_0(X_1=x_1)$. A cada possível *posteriori* $\xi_0(X_1=x_1)$, contudo, corresponderá um particular conjunto tangente a Θ_0, diretamente dependente de x_1 e podendo ser representado por $T^*(x_1)$.

Como a valores diferentes que x_1 puder assumir correspondem conjuntos tangentes $T^*(x_1)$ potencialmente diferentes, o cálculo analítico de $E[\rho_0(\xi_0(X_1))]$ exigiria a análise de todos os conjuntos tangentes possíveis e de suas respectivas evidências, o que é impraticável, uma vez que a simples obtenção analítica do conjunto tangente e da evidência para uma única distribuição particular já é por si só algo suficientemente complexo e nem sempre realizável.

Assim, sendo o procedimento analítico padrão impraticável, torna-se necessário buscar formas alternativas de se estender o FBST para o contexto seqüencial. Sugere-se, então, como

alternativa às abordagens puramente teóricas, uma abordagem algorítmica, em que a difícil obtenção direta de $E[\rho_0(\xi_0(X_1))]$ é substituída por sua estimação por meio de simulações.

4.3. Abordagem via simulações

4.3.1. Algoritmo básico

Conforme verificado na Seção 3.2, a *posteriori* $\xi_0(x_1)$, seus respectivos conjunto tangente e medida de evidência e, em última instância, o risco $E[\rho_0(\xi_0(X_1))]$ dependem direta e unicamente do valor amostrado x_1.

Além disso, para as distribuições da família exponencial, pode-se fazer uso das classes de distribuições conjugadas, o que torna fácil obter diretamente os (hiper)parâmetros da *posteriori* a partir dos (hiper)parâmetros da *priori* e do valor observado.

Assim, pode-se inicialmente formular o seguinte algoritmo padrão para a execução seqüencial do FBST:

Algoritmo 1 (básico):

0. Calcule $\rho_0 = \min\{a\, Ev(\Theta_0, \xi_0)\,; b + c\,(1 - Ev(\Theta_0, \xi_0))\}$.
1. Inicialize o contador de simulações ($i = 0$).
2. Faça $i = i + 1$.
   3. Gere $\theta_i \sim \xi_0(\theta)$.
   4. Gere $x_i \sim f(x \mid \theta_i)$.
   5. Calcule os (hiper)parâmetros da *posteriori* $\xi_i = \xi_0(X_1 = x_i)$.
   6. Calcule a evidência $Ev_i = Ev(\Theta_0, \xi_i)$.
   7. Calcule o risco $\rho_i = \min\{a\, Ev_i\,; b + c\,(1 - Ev_i)\}$.
8. Se $i < N$, volte para o passo 2; se não, siga para o passo 9.
9. Calcule $\overline{\rho}_N = \frac{1}{N}\sum_{i=1}^{N} \rho_i$.
10. Se $\overline{\rho}_N + C < \rho_0$, então amostre mais uma observação; caso contrário, encerre o processo de amostragem.

Algumas observações devem ser tecidas acerca desse algoritmo:

a) É fácil ver que $E[\overline{\rho}_N] = E[\rho_0(\xi_0(X_1))]$, ou seja, que esse algoritmo fornece, efetivamente, um estimador não-viciado para o risco do prosseguimento da amostragem.

b) Além disso, é fácil ver que $Var[\overline{\rho}_N] = \frac{Var[\rho_0(\xi_0(X_1))]}{N}$ e, por conseguinte, que, quanto maior for o valor de *N*, menor será a oscilação de $\overline{\rho}_N$ em torno de $E[\rho_0(\xi_0(X_1))]$.

Isso sugere que, em lugar de se estabelecer antecipadamente um número arbitrário *N* de iterações, o critério de parada do passo 8 pode ser substituído pelos passos abaixo, de forma a garantir que a estimativa de $E[\rho_0(\xi_0(X_1))]$ seja tão precisa quanto se queira:

8.1. Calcule $\overline{\rho}_i = \frac{1}{i}\sum_{j=1}^{i}\rho_j$.

8.2. Se $\left|\frac{\overline{\rho}_i - \overline{\rho}_{i-1}}{\overline{\rho}_{i-1}}\right| > \varepsilon$, volte para o passo 2; se não, siga para o passo 9.

c) O cálculo da evidência, efetuado no passo 6, pode ser realizado segundo os algoritmos implementados por Faria Jr. (2006). Esses algoritmos também fornecem estimativas do erro cometido na computação da Evidência, calculadas por Faria Jr. (2006), com base em Stern e Zacks (2002) e Stern (2003).

d) Para situações em que as distribuições (*priori/posteriori* e verossimilhança) não sejam conjugadas, pode-se substituir o passo 5 por rotinas de simulação de *posterioris* como, por exemplo, o ABC - *Aproximated Bayesian Computing* (Marjoram *et al*, 2003).

4.3.2. Algoritmo expandido

O algoritmo básico, apresentado na Subseção anterior, tem como aspecto potencialmente inconveniente o fato de confrontar apenas o risco da decisão ótima imediata com o risco da decisão ótima tomada após ser amostrada mais uma observação. Isso significa que, após cada sucessiva observação ser amostrada, o algoritmo deverá ser novamente executado, o que pode acarretar custos adicionais (i.e. gastos financeiros, tempo consumido, trabalho executado etc.) decorrentes das sucessivas interrupções e reinícios do processo amostral.

Não se vai entrar aqui no estudo da quantificação e do tratamento matemático desses custos adicionais, mas estes certamente podem ser reduzidos se o algoritmo for expandido de forma a confrontar também os riscos das decisões ótimas tomadas após a amostragem de mais *k* observações, para diversos valores de *k*:

Algoritmo 2 (expandido):

0.1. Calcule $\rho_0 = \min\{a\,Ev(\Theta_0, \xi_0)\,; b + c(1 - Ev(\Theta_0, \xi_0))\}$.

0.2. Calcule $m_0 = \rho_0$

1. Inicialize o tamanho de amostra ($k = 0$).

2. Faça $k = k + 1$

    3. Inicialize o contador de simulações ($i = 0$) .

    4. Faça $i = i + 1$.

        5. Gere $\theta_i \sim \xi_0(\theta)$.

        6. Gere $x_{i1}, \ldots, x_{ik} \sim f(x \mid \theta_i)$.

        7. Calcule os (hiper)parâmetros da *posteriori* $\xi_i = \xi_0(X_{i1} = x_{i1}, \ldots, X_{ik} = x_{ik})$.

        8. Calcule a evidência $Ev_i = Ev(\Theta_0, \xi_i)$.

        9. Calcule o risco $\rho_i = \min\{a\,Ev_i\,; b + c(1 - Ev_i)\}$.

10. Calcule $\overline{\rho}_i = \frac{1}{i}\sum_{j=1}^{i} \rho_j$.

11. Se $\left|\frac{\overline{\rho}_i - \overline{\rho}_{i-1}}{\overline{\rho}_{i-1}}\right| > \varepsilon$, volte para o passo 4; se não, siga para o passo 12.

12. Calcule $\mu_k = \overline{\rho}_i$.

13. Se $\mu_k + kC \geq m_{k-1}$, então encerre o algoritmo e amostre exatas $k-1$ observações; caso contrário, siga para o passo 14;

14. Calcule $m_k = \min\{m_{k-1}\,; \mu_k + kC\}$ e volte para o passo 2.

4.3.3. Valores a, b e c e possíveis aperfeiçoamentos ao algoritmo

Os valores *a*, *b* e *c* empregados nos algoritmos apresentados nas Subseções anteriores carecem de significado prático, o que inviabiliza seu anúncio num processo de tomada de decisão. Esse fato, por conseguinte, enseja a formulação de diferentes representações desses valores, de forma a que possam ser associados a algum padrão que possa ser avaliado e anunciado pelo tomador de decisão.

Um possível aperfeiçoamento do algoritmo, no sentido da substituição dos parâmetros *a*, *b* e *c* por valores de mais fáceis interpretação e anúncio, se baseia pela comparação do critério para aceitação de H_0 estabelecido na Seção 2.2 com um valor previamente arbitrado pelo tomador de decisão. Supondo que o tomador de decisão estabeleça um valor α tal que H_0 será aceita se e somente se $Ev(\Theta_0, x) > \alpha$, isso naturalmente ensejará a igualdade

$$\boxed{\frac{b+c}{a+c} = \alpha}$$

Essa igualdade envolve as três variáveis *a*, *b* e *c* e, por esse motivo, é insuficiente para substituir <u>todas</u> essas variáveis por valores com intepretação ou significado prático.

Pode-se, porém, estabelecer essa comparação não com os valores *a*, *b* e *c* originais mas com valores a', b' e c' definidos em função dos parâmetros originais:

$$\begin{cases} a' = a - b \\ b' = 0 \\ c' = b + c \end{cases}$$

Essa escolha de a', b' e c' não altera a essência do problema, pois:

a) O critério para aceitação de H_0 permanece o mesmo:

$$\frac{b'+c'}{a'+c'} = \frac{0+(b+c)}{(a-b)+(b+c)} = \frac{b+c}{a+c}$$

b) Os riscos de cada decisão possível correspondem exatamente aos riscos originais, subtraindo-se apenas um valor constante $bEv(\Theta_0, x)$:

$$E_\xi[LP(rejeitar\ H_0, \theta) \mid x] = a' Ev(\Theta_0, x) = (a-b)Ev(\Theta_0, x) = \\ = (aEv(\Theta_0, x)) - (bEv(\Theta_0, x))$$

$$E_\xi[LP(aceitar\ H_0, \theta) \mid x] = b' + c'(1 - Ev(\Theta_0, x)) = (b+c)(1 - Ev(\Theta_0, x))` = \\ = (b + c(1 - Ev(\Theta_0, x))) - (bEv(\Theta_0, x))$$

c) Os riscos das situações-limite são completamente intuitivos (talvez até mais intuitivos que os riscos expressos em função dos parâmetros originais), pois:

- $Ev(\Theta_0, x) = 0 \Rightarrow E_\xi[LP(rejeitar\ H_0, \theta) \mid x] = a' Ev(\Theta_0, x) = 0$, o que equivale a dizer que, tendo "certeza absoluta" (evidência igual a zero) de que H_0 é falsa, o risco que se corre ao rejeitá-la é nulo;
- $Ev(\Theta_0, x) = 1 \Rightarrow E_\xi[LP(aceitar\ H_0, \theta) \mid x] = c'(1 - Ev(\Theta_0, x)) = 0$, que equivale a dizer que, tendo "certeza absoluta" (evidência igual a um) de que H_0 é verdadeira, o risco que se corre ao aceitá-la é nulo.

Nessa representação, então, o critério para aceitação/rejeição de H_0 conduz à igualdade:

$$\boxed{\frac{c'}{a' + c'} = \alpha} \quad .$$

Considerando-se, então, os riscos (de aceitação e rejeição de H_0) expressos em função de a', $b'(=0)$ e c', tem-se que o risco da decisão ótima será igual a:

$$\rho^* = \begin{cases} a' Ev(\Theta_0, x) & \text{se } H_0 \text{ for rejeitada} \quad (i.e.\ se\ Ev(\Theta_0, x) \le \alpha) \\ c'(1 - Ev(\Theta_0, x)) & \text{se } H_0 \text{ for aceita} \quad (i.e.\ se\ Ev(\Theta_0, x) \ge \alpha) \end{cases} ,$$

de onde pode-se verificar facilmente que $\boxed{0 \le \rho^* \le c'(1-\alpha)}$ ou, equivalentemente, que $\boxed{0 \le \rho^* \le a'\alpha}$.

Logo, pode-se interpretar $R^* = c'(1-\alpha) = a'\alpha$ como o maior risco possível que se corre ao se tomar a decisão ótima. O tomador de decisão, presumivelmente familiarizado com o contexto do problema, deverá ser capaz de enunciar um valor para R^*, "o maior prejuízo que você poderá sofrer se tomar a decisão ótima". Assim, de posse desse valor, pode-se, por fim, substituir os parâmetros originais *a*, *b* e *c* por:

$$\boxed{\begin{cases} a' = \dfrac{R^*}{\alpha} \\ b' = 0 \\ c' = \dfrac{R^*}{1-\alpha} \end{cases}}$$

e o Algoritmo 2 pode ser adaptado da seguinte forma:

Algoritmo 3 (expandido adaptado):

0.1. Estabeleça valores α e R^* e calcule $a = \dfrac{R^*}{\alpha}$ e $c = \dfrac{R^*}{1-\alpha}$.

0.2. Calcule $\rho_0 = \min\{a\, Ev(\Theta_0, \xi_0)\,; c(1 - Ev(\Theta_0, \xi_0))\}$.

0.3. Calcule $m_0 = \rho_0$

1. Inicialize o tamanho de amostra ($k = 0$).

2. Faça $k = k + 1$

    3. Inicialize o contador de simulações ($i = 0$).

    4. Faça $i = i + 1$.

        5. Gere $\theta_i \sim \xi_0(\theta)$.

        6. Gere $x_{i1}, \ldots, x_{ik} \sim f(x \mid \theta_i)$.

        7. Calcule os (hiper) parâmetros da *posteriori* $\xi_i = \xi_0(X_{i1} = x_{i1}, \ldots, X_{ik} = x_{ik})$.

        8. Calcule a evidência $Ev_i = Ev(\Theta_0, \xi_i)$.

        9. Calcule o risco $\rho_i = \min\{a\, Ev_i\,; c(1 - Ev_i)\}$.

        10. Calcule $\overline{\rho}_i = \dfrac{1}{i}\sum_{j=1}^{i} \rho_j$.

    11. Se $\left|\dfrac{\overline{\rho}_i - \overline{\rho}_{i-1}}{\overline{\rho}_{i-1}}\right| > \varepsilon$, volte para o passo 4; se não, siga para o passo 12.

    12. Calcule $\mu_k = \overline{\rho}_i$.

13. Se $\mu_k + kC \geq m_{k-1}$, então encerre o algoritmo e amostre exatas $k-1$ observações; caso contrário, siga para o passo 14;

14. Calcule $m_k = \min\{m_{k-1}\,; \mu_k + kC\}$ e volte para o passo 2.

É importante observar que, para que o algoritmo funcione adequadamente, os valores de R^* e C devem ser expressos na mesma "unidade de medida", de forma a que a soma $\mu_k + kC$ e a comparação $\min\{m_{k-1}\,; \mu_k + kC\}$ façam pleno sentido.

Cabe citar, também, que essa não é necessariamente a única maneira possível de se expressar os parâmetros *a*, *b* e *c* em função de grandezas concretas e/ou interpretáveis. É, portanto, um tópico importante a ser explorado a procura ou formulação de outra(s) reparametrização(ões) que possa(m) fornecer alternativas práticas e interpretáveis ao vetor de parâmetros (a, b, c).

Capítulo 5
Aplicações

5.1. Teste de independência entre variáveis Poisson

Dadas duas variáveis aleatórias com distribuição de Poisson, deseja-se testar a hipótese de que essas variáveis sejam independentes contra a hipótese de que possuam uma distribuição conjunta de Holgate (1964).

<u>*Definição:*</u> Considerando-se três variáveis aleatórias independentes $P_1 \sim Poisson(\lambda_1)$, $P_2 \sim Poisson(\lambda_2)$ e $P_3 \sim Poisson(\lambda_3)$, então o vetor $(X,Y) = (P_1 + P_3, P_2 + P_3)$ é dito possuir uma Distribuição de Holgate com parâmetros $\theta_1 = \lambda_1 + \lambda_3$, $\theta_2 = \lambda_2 + \lambda_3$ e $\theta_3 = \lambda_3$ e função de probabilidade

$$P(X = x, Y = y) = e^{-(\theta_1+\theta_2-\theta_3)} \sum_{i=0}^{\min(x,y)} \frac{(\theta_1 - \theta_3)^{x-i} (\theta_2 - \theta_3)^{y-i} \theta_3^i}{(x-i)!(y-i)!i!} \quad .$$

Nesse caso, tem-se $E[X] = \theta_1$, $E[Y] = \theta_2$ e $Cov(X,Y) = \theta_3$. ■

Considerando-se, então, o espaço paramétrico $\Theta = \{\theta_1, \theta_2 \geq 0 ; 0 \leq \theta_3 \leq \min\{\theta_1, \theta_2\}\}$, Stern e Zacks (2002) formularam o FBST para a hipótese $\Theta_0 = \{\theta_1, \theta_2 \geq 0 ; \theta_3 = 0\}$ da seguinte forma:

Dada uma amostra $W = (W_1, W_2, \ldots, W_n)$, com $W_i = (X_i, Y_i)$, definem-se as funções de verossimilhança e evidência:

$$L_i(\theta, W_i) = P(W_i \mid \theta), \text{ onde } \theta = (\theta_1, \theta_2, \theta_3)$$

$$L(\theta, W) = \prod_{i=1}^{n} L_i(\theta_i, W_i)$$

$$Ev(W) = \frac{\int_{\theta^*} L(\theta, W) d\theta}{\int_{\theta} L(\theta, W) d\theta}, \text{ onde } \theta^* \text{ é o conjunto tangente a } H_0.$$

Essa evidência é, então, estimada via *Monte Carlo Importance Sampling* (Anderson, 1999), utilizando-se como função de importância uma densidade $g(\theta)$, positiva em Θ:

$$Ev(W) = \frac{\int_{\theta} Z_g^*(\theta, W) g(\theta) d\theta}{\int_{\theta} Z_g(\theta, W) g(\theta) d\theta},$$

onde $Z_g(\theta, W) = \dfrac{L(\theta, W)}{g(\theta)}$, $Z_g^*(\theta, W) = I^*(\theta, W) Z_g(\theta, W)$,

$I^*(\theta, W) = \mathbf{1}\{L(\theta, W) \geq \ell^*\}$ e ℓ^* é o valor máximo de $L(\theta, W)$ sob H_0.

Como X e Y são independentes sob H_0 e é sabido que o estimador de máxima verossimilhança para o parâmetro de uma Poisson univariada é a sua média amostral, conclui-se que $\ell^* = L((\overline{X},\overline{Y},0),W)$.

Então, a estimativa da Evidência será dada por

$$\hat{E}v_{g,m}(W) = \frac{\sum_{i=1}^{m} Z_g^*(\theta_{i,\bullet},W)}{\sum_{i=1}^{m} Z_g(\theta_{i,\bullet},W)},$$

onde $\theta_{1,\bullet}, \theta_{2,\bullet}, \ldots, \theta_{m,\bullet}$ são valores i.i.d. gerados aleatoriamente a partir da função de importância $g(\theta)$.

Por ser uma distribuição conjugada à de Poisson, escolhe-se a distribuição Gama como geradora de θ_1 e θ_2, além de uma Uniforme$[0;\min(\theta_1,\theta_2)]$ como geradora de θ_3:

$$g(\theta) = G(\theta_1 \mid \alpha_1,\beta_1)G(\theta_2 \mid \alpha_2,\beta_2)\frac{\mathbf{1}\{\theta_3 \le \min(\theta_1,\theta_2)\}}{\min(\theta_1,\theta_2)},$$

onde $G(\theta \mid \alpha,\beta) = \frac{\theta^{\alpha-1}e^{-\beta\theta}\beta^{\alpha}}{\Gamma(\alpha)}$.

Por fim, utiliza-se como parâmetros para a função *G* os valores da *posteriori* conjugada:

$\alpha_1 = n\overline{X}$, $\beta_1 = n$, $\alpha_2 = n\overline{Y}$ e $\beta_2 = n$.

Assim, aplicando-se a esse teste o exposto na seção 4.3, pode-se formular o seguinte algoritmo para a execução seqüencial do teste de independência entre variáveis Poisson:

Algoritmo 4 (Independência de Poissons):

0.1. Calcule $Ev_0 = \hat{E}v_{g,m}(W_1,W_2,\ldots,W_n)$

0.2. Estabeleça valores α e R^* e calcule $a = \frac{R^*}{\alpha}$ e $c = \frac{R^*}{1-\alpha}$.

0.3. Calcule $\rho_0 = \min\{a\,Ev_0\,; c(1-Ev_0)\}$.

0.4. Calcule $m_0 = \rho_0$

0.5. Calcule $\alpha_1 = n\overline{X}$, $\beta_1 = n$, $\alpha_2 = n\overline{Y}$ e $\beta_2 = n$.

1. Inicialize o tamanho de amostra ($k = 0$).

2. Faça $k = k+1$

    3. Inicialize o contador de simulações ($i = 0$) .

    4. Faça $i = i+1$.

        5. Gere $\theta_i = (\theta_{i1},\theta_{i2},\theta_{i3}) \sim g(\alpha_1,\beta_1,\alpha_2,\beta_2)$.

        6. Gere $w_{i1},\ldots,w_{ik} \sim Holgate(\theta_{i1},\theta_{i2},\theta_{i3})$.

7. Calcule a evidência $Ev_i = \hat{E}v_{g,m}(W_1, W_2, \ldots, W_n, w_{i1}, w_{i2}, \ldots, w_{ik})$.

8. Calcule o risco $\rho_i = \min\{aEv_i\ ; c(1-Ev_i)\}$.

9. Calcule $\overline{\rho}_i = \frac{1}{i}\sum_{j=1}^{i}\rho_j$.

10. Se $\left|\frac{\overline{\rho}_i - \overline{\rho}_{i-1}}{\overline{\rho}_{i-1}}\right| > \varepsilon$, volte para o passo 4; se não, siga para o passo 12.

11. Calcule $\mu_k = \overline{\rho}_i$.

12. Se $\mu_k + kC \geq m_{k-1}$, então encerre o algoritmo e amostre exatas $k-1$ observações; caso contrário, siga para o passo 13;

13. Calcule $m_k = \min\{m_{k-1}\ ; \mu_k + kC\}$ e volte para o passo 2.

5.2. Testes em tabelas 2x2

Dados dois tratamentos e uma amostra de indivíduos submetidos a ambos, deseja-se testar hipóteses relativas às distribuições marginais e conjunta dos dados. Pode-se representar essa estrutura em uma tabela 2x2 como segue:

		Tratamento I		
		Sucessos	Fracassos	
Tratamento II	Sucessos	p_{11}	p_{12}	p_y
	Fracassos	p_{21}	p_{22}	$1-p_y$
		p_x	$1-p_x$	1

Trata-se, então, de uma variável-resposta bivariada (*X*,*Y*), onde:

X = 1 se for observado um sucesso e 0 se um fracasso no Tratamento I;
Y = 1 se for observado um sucesso e 0 se um fracasso no Tratamento II;

e, portanto,

$X \sim Bernoulli(p_x)$, onde p_x = P(sucesso no Tratamento I);
$Y \sim Bernoulli(p_y)$, onde p_y = P(sucesso no Tratamento II).

Aqui, Tratamento I e Tratamento II podem ser, dentre outros exemplos:

- duas provas a que alunos tenham sido submetidos (e nesse caso os sucessos e fracassos seriam as aprovações e reprovações de cada aluno em cada prova);
- duas afirmações apresentadas a um grupo de entrevistados (nesse caso os sucessos e fracassos seriam as concordâncias e discordâncias de cada entrevistado em relação a cada afirmação);
- intenções de voto em um candidato antes e depois de determinado evento (nesse caso, os sucessos e fracassos seriam as intenções de votar e de não votar nesse candidato em cada um dos momentos avaliados).

Assim, uma amostra $\mathbf{W} = (w_{11}, w_{12}, w_{21}, w_{22})$, formada por N indivíduos e representada na tabela a seguir, segue uma distribuição Multinomial com parâmetros $(N, p_{11}, p_{12}, p_{21}, p_{22})$ e, dependendo do experimento realizado, duas hipóteses distintas podem ser de interesse: $H_{0a} : p_{12} = p_{21}$ ou $H_{0b} : p_{11} \cdot p_{22} = p_{12} \cdot p_{21}$.

		Tratamento I		
		Sucessos	Fracassos	
Tratamento II	Sucessos	W_{11}	W_{12}	Y
	Fracassos	W_{21}	W_{22}	$N - Y$
		X	$N - X$	N

Aqui, W_{ij} é a quantidade de observações nas quais $X = 2 - i$ e $Y = 2 - j$.

Então, as hipóteses citadas podem ser interpretadas, em termos do contexto do experimento (e invocando os três exemplos de tratamentos anteriormente citados), das seguintes formas:

$H_{0a} : p_{12} = p_{21}$ equivale a $H_{0a} : E[X] = p_x = p_y = E[Y]$, que equivale a:
- a perícia/exigência das duas provas é igual;
- o apoio da população às duas afirmações é igual;
- as intenções de voto antes e depois do evento são iguais;
- as moedas *X* e *Y* têm probabilidades iguais de sair cara.

$H_{0b} : p_{11} \cdot p_{22} = p_{12} \cdot p_{21}$ equivale a $H_{0b} : Cov(X, Y) = 0$, que equivale a:
- não existe dependência entre as aprovações na prova 1 e na prova 2;
- não existe dependência entre os apoios à afirmação 1 e à afirmação 2;
- não existe dependência entre as intenções de voto antes e depois do evento;
- não existe dependência entre os resultados das moedas *X* e *Y*.

Pereira *et al* (2008) aplicam a H_{0a} o FBST padrão para amostras multinomiais com *priori* e *posteriori* Dirichlet, ou seja:

$$\theta = (p_{11}, p_{12}, p_{21}, p_{22}) \sim Dirichlet(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$$
$$\mathbf{W} \mid \theta \sim Multinomial(p_{11}, p_{12}, p_{21}, p_{22})$$
$$\theta \mid \mathbf{W} \sim Dirichlet(\alpha_1 + W_{11}, \alpha_2 + W_{12}, \alpha_3 + W_{21}, \alpha_4 + W_{22}) \quad ,$$

onde $\mathbf{W} = (w_{11}, w_{12}, w_{21}, w_{22})$ é o vetor de observações, definido na tabela mais acima.

Nesse teste, o ponto de máximo da *posteriori* sob a hipótese nula (θ^*), o conjunto tangente e a medida de Evidência são obtidos rigorosamente da forma descrita na seção 2.1. Como esse arcabouço é igualmente válido para o teste de H_{0b}, pode-se formular o seguinte algoritmo para a execução seqüencial de qualquer dos dois testes para tabelas 2x2, diretamente a partir daquele construído na Subseção 4.3.3:

Algoritmo 5 (tabelas 2x2):

0.1. Estabeleça valores α e R^* e calcule $a = \frac{R^*}{\alpha}$ e $c = \frac{R^*}{1-\alpha}$.
0.2. Calcule $\rho_0 = \min\{a\,Ev(\Theta_0, \xi_0)\,; c(1 - Ev(\Theta_0, \xi_0))\}$.
0.3. Calcule $m_0 = \rho_0$
1. Inicialize o tamanho de amostra ($k = 0$).
2. Faça $k = k + 1$
    3. Inicialize o contador de simulações ($i = 0$).
    4. Faça $i = i + 1$.
        5. Gere $\theta_i = (p_{11i}, p_{12i}, p_{21i}, p_{22i}) \sim Dirichlet(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$.
        6. Gere $\mathbf{W}_{i1}, \ldots, \mathbf{W}_{ik} \sim Multinomial(p_{11i}, p_{12i}, p_{21i}, p_{22i})$.
        7. Calcule os (hiper)parâmetros da *posteriori Dirichlet* $\xi_i(\theta \mid \mathbf{W}_{i1}, \ldots, \mathbf{W}_{ik})$
        8. Calcule a evidência $Ev_i = Ev(\Theta_0, \xi_i)$.
        9. Calcule o risco $\rho_i = \min\{a\,Ev_i\,; c(1 - Ev_i)\}$.
        10. Calcule $\overline{\rho}_i = \frac{1}{i}\sum_{j=1}^{i} \rho_j$.
    11. Se $\left|\frac{\overline{\rho}_i - \overline{\rho}_{i-1}}{\overline{\rho}_{i-1}}\right| > \varepsilon$, volte para o passo 2; se não, siga para o passo 12.
    12. Calcule $\mu_k = \overline{\rho}_i$.
13. Se $\mu_k + kC \geq m_{k-1}$, então encerre o algoritmo e amostre exatas $k-1$ observações; caso contrário, siga para o passo 14;
14. Calcule $m_k = \min\{m_{k-1}\,; \mu_k + kC\}$ e volte para o passo 2.

5.3. O Problema de Behrens-Fisher

Dadas duas variáveis aleatórias independentes *X* e *Y* com distribuição Normal com médias μ_1 e μ_2 e variâncias σ_1^2 e σ_2^2 desconhecidas (e supostas diferentes), o problema de Behrens-Fisher consiste em testar a hipótese $H_0 : \mu_1 = \mu_2$.

Madruga (2002), Madruga *et al* (2003) e Campos *et al* (2005) constroem o FBST para essa hipótese nula valendo-se de *prioris* impróprias para o vetor $\theta = (\mu_1, \mu_2, \sigma_1^2, \sigma_2^2)$. É possível, porém, efetuar esse teste utilizando-se unicamente *prioris* próprias de famílias conjugadas, conforme desenvolvimento a seguir.

Sejam $\mathbf{X} = (x_1, x_2, \ldots, x_n)$ e $\mathbf{Y} = (y_1, y_2, \ldots, y_m)$ valores amostrados de *X* e *Y*. Então, por serem *X* e *Y* independentes, a distribuição (verossimilhança) conjunta de **X** e **Y** pode evidentemente ser fatorada no produto das respectivas distribuições marginais:

$$f(\mathbf{X}, \mathbf{Y} \mid \mu_1, \mu_2, \sigma_1^2, \sigma_2^2) = f(\mathbf{X} \mid \mu_1, \sigma_1^2) \cdot f(\mathbf{Y} \mid \mu_2, \sigma_2^2).$$

Analogamente, pode-se supor uma *priori* conjunta fatorável no produto de duas distribuições “marginais”:

$$\pi(\mu_1,\mu_2,\sigma_1^2,\sigma_2^2)=\pi(\mu_1,\sigma_1^2)\cdot\pi(\mu_2,\sigma_2^2)$$

È importante frisar que essa fatoração depende diretamente da suposição de independência entre os vetores (μ_1,σ_1^2) e (μ_2,σ_2^2). Admitindo-se aceitas essa suposição e a fatoração acima, pode-se, de forma similar ao realizado com a *priori*, fatorar a *posteriori* conjunta no produto de duas *posterioris* “marginais” da seguinte forma:

$$\begin{aligned}\pi(\mu_1,\mu_2,\sigma_1^2,\sigma_2^2\mid\mathbf{X},\mathbf{Y}) &\propto \pi(\mu_1,\mu_2,\sigma_1^2,\sigma_2^2)\cdot f(\mathbf{X},\mathbf{Y}\mid\mu_1,\mu_2,\sigma_1^2,\sigma_2^2)= \\ &= \pi(\mu_1,\sigma_1^2)\cdot\pi(\mu_2,\sigma_2^2)\cdot f(\mathbf{X}\mid\mu_1,\sigma_1^2)\cdot f(\mathbf{Y}\mid\mu_2,\sigma_2^2)= \\ &= \pi(\mu_1,\sigma_1^2)\cdot f(\mathbf{X}\mid\mu_1,\sigma_1^2)\cdot\pi(\mu_2,\sigma_2^2)\cdot f(\mathbf{Y}\mid\mu_2,\sigma_2^2)\propto \\ &\propto \pi(\mu_1,\sigma_1^2\mid\mathbf{X})\cdot\pi(\mu_2,\sigma_2^2\mid\mathbf{Y})\end{aligned}$$

Logo, como a distribuição Normal com média e variância desconhecidas tem como conjugada a Normal-Gama, as fatorações acima permitem que se considerem, no presente caso, a verossimilhança como um produto de Normais independentes e as distribuições *a priori* e *a posteriori* como produtos de Normais-Gama independentes.

Assim, pode-se formular para esse teste o algoritmo abaixo, diretamente adaptado do algoritmo básico da Subseção 4.3.3. Com adaptações simples dos passos 5 a 7, também é possível aplicar esse algoritmo a outras *prioris*, como as impróprias utilizadas por Madruga (2002), Madruga *et al* (2003) e Campos *et al* (2005).

Algoritmo 6 (Behrens-Fisher):

0.1. Estabeleça valores α e R^* e calcule $a=\dfrac{R^*}{\alpha}$ e $c=\dfrac{R^*}{1-\alpha}$.

0.2. Calcule $\rho_0=\min\{a\,Ev(\Theta_0,\xi_0);c(1-Ev(\Theta_0,\xi_0))\}$.

0.3. Calcule $m_0=\rho_0$

1. Inicialize o tamanho de amostra ($k=0$).

2. Faça $k=k+1$

3. Inicialize o contador de simulações ($i=0$).

4. Faça $i=i+1$.

5. Gere $(M_j,p_j)\sim Normal\text{-}Gama(\mu_j,\tau_j,\alpha_j,\beta_j)$ e $v_j=\dfrac{1}{p_j}$ $(j=1,2)$.

6. Gere $x_{i1},\ldots,x_{ik}\sim N(M_1,v_1)$ e $y_{i1},\ldots,y_{ik}\sim N(M_2,v_2)$.

7. Calcule os (hiper)parâmetros das *posterioris Normal-Gama* $\xi_{ix}(M_1,p_1\mid x_{i1},\ldots,x_{ik})$ e $\xi_{iy}(M_2,p_2\mid y_{i1},\ldots,y_{ik})$.

8. Calcule a evidência $Ev_i=Ev(\Theta_0,\xi_{ix}\cdot\xi_{iy})$.

9. Calcule o risco $\rho_i=\min\{a\,Ev_i;c(1-Ev_i)\}$.

10. Calcule $\overline{\rho}_i=\dfrac{1}{i}\sum_{j=1}^{i}\rho_j$.

11. Se $\left|\frac{\overline{\rho}_i - \overline{\rho}_{i-1}}{\overline{\rho}_{i-1}}\right| > \varepsilon$, volte para o passo 2; se não, siga para o passo 12.

12. Calcule $\mu_k = \overline{\rho}_i$.

13. Se $\mu_k + kC \geq m_{k-1}$, então encerre o algoritmo e amostre exatas $k-1$ observações; caso contrário, siga para o passo 14;

14. Calcule $m_k = \min\{m_{k-1}; \mu_k + kC\}$ e volte para o passo 2.

Capítulo 6
Execução do FBST Seqüencial no software *R*

6.1. Descrição dos programas

Foi desenvolvida uma biblioteca de comandos para execução do FBST Seqüencial (para algumas situações específicas) no *software* R. Foram também adaptadas e atualizadas duas bibliotecas pré-existentes (Faria Jr., 2006) cujo conteúdo é necessário à implementação do FBST Seqüencial.

Esse conjunto de bibliotecas, cujos códigos-fonte se encontram no Apêndice A e também disponível para *download* na página http://www.chancedegol.com.br/sfbst/sfbst.htm, consiste dos seguintes elementos:

a) Biblioteca *yacas0.R*, originalmente escrita por Faria Jr. (2006), que inclui:

i) Atualização e ajustes dos comandos e funções para interação entre os *softwares* R, Yacas e Algencan.

O Yacas (http://yacas.sourceforge.net) é um *software* voltado para problemas de álgebra simbólica e capaz de realizar operações de cálculo diferencial e integral, enquanto o Algencan (http://www.ime.usp.br/~egbirgin/tango) é uma rotina desenvolvida pelo Projeto Tango com o objetivo de resolver problemas de otimização de funções sujeitas a restrições de igualdade e/ou de desigualdade.

As rotinas originalmente implementadas por Faria Jr. (2006) se baseavam nas chamadas externas aos programas Yacas e Algencan, enquanto as rotinas aqui apresentadas têm como principal adaptação o emprego das hoje existentes bibliotecas *Ryacas* e *alabama*, o que permite que todo o processo seja executado internamente no ambiente R.

O programa Algencan é baseado no mesmo algoritmo de otimização (Lagrangeano Aumentado (Lange, 2004, Madsen *et al*, 2004)), empregado na função *auglag* da biblioteca *alabama*, o que comprova a adequação da utilização dessa biblioteca em substituição à chamada externa àquele programa.

ii) Atualização e ajustes dos comandos e funções responsáveis pela tradução entre as notações (ou "linguagens") em que o *software* R lê e manipula funções e seus respectivos argumentos numéricos, vetoriais ou matriciais.

b) Biblioteca *fbst.R*, originalmente escrita por Faria Jr. (2006), que inclui:

i) Atualização e ajustes dos comandos e funções utilizadas para o cálculo da evidência contrária à hipótese nula, executado por meio do algoritmo de *Monte Carlo Importance Sampling* (Anderson, 1999, Faria Jr., 2006).

ii) Atualização e ajustes dos comandos e funções responsáveis pela comunicação e interação entre as rotinas de maximização da *posteriori* e de cálculo da evidência contrária à hipótese nula.

Todos os comandos e funções dessa biblioteca foram modificados basicamente em função das alterações e atualizações feitas aos comandos e funções da biblioteca *yacas0.R*.

c) Biblioteca *sfbst.R*, que inclui rotinas para a execução do FBST Seqüencial aplicado aos casos mais usuais de distribuições (verossimilhança e *priori*/*posteriori*) conjugadas uniparamétricas, bem como aos principais casos multiparamétricos univariados (Fink, 1997):

Distribuição de Verossimilhança	***Priori*/*Posteriori* Conjugada**	**Função/Comando da biblioteca *sfbst.R***
Bernoulli Binomial Geométrica Binomial Negativa	Beta	sfbst.betabin
Poisson	Gama	sfbst.gamapois
Exponencial	Gama	sfbst.gamaexp
Normal (variância conhecida)	Normal	sfbst.normnorm
Normal (média conhecida)	Gama	sfbst.gamanorm
Multinomial	Dirichlet	sfbst.dirimult
Normal (média e variância desconhecidas)	Normal-Gama	sfbst.normgama

d) Biblioteca *sfbstapp.R*, que inclui rotinas específicas para a execução do FBST Seqüencial aplicado ao testes de Independência entre variáveis Poisson (tratado na Seção 5.1) e ao Problema de Behrens-Fisher (abordado na Seção 5.3).

Os testes para tabelas 2x2 (considerados na Seção 5.2) são executados por meio da função sfbst.dirimult, da biblioteca *sfbst.R*, conforme ilustrado no Exemplo 3 da Seção a seguir.

6.2. Exemplos

As bibliotecas foram desenvolvidas e testadas e os exemplos a seguir foram rodados num computador com as seguintes configurações:

- Desktop PC, Intel Core2Duo, 2.93 GHz, 2.00 Gb de RAM;
- Sistema Operacional Windows XP Professional Versão 2002, *Service Pack* 3;
- *Software* R versão 2.12.2

Também foram efetuados testes num computador PC com processador Intel Core2Duo, 2.33 GHz, 4.00 Gb de RAM, sistema operacional Windows Vista Home Premium Versão 2007, *Service Pack* 2 e *software* R versões 2.13.0 e x64 2.13.0.

São apresentados, a seguir, alguns exemplos que ilustram a execução dos comandos desenvolvidos:

<u>Exemplo 1</u>: Estimação do parâmetro de uma distribuição Exponencial (λ).

Considere-se que, após a realização de uma série de observações, a distribuição *a posteriori* para λ seja uma Gama (11, 12). A hipótese que se deseja testar é $H_0 : \lambda = 1$ e os comandos abaixo produzem os resultados a seguir:

```
#####################################################################
# Limites da região do espaço paramétrico onde será buscado o ponto
# de máximo da posteriori sob H0
bounds <- list(lower = 0.001, upper = 40)

# Hipótese nula, escrita sob a forma de restrição de igualdade
const <- "x1 - 1"

# Ponto inicial para o procedimento de otimização da posteriori
init <- 0.9

# Função de importância para o Importance Sampling
sampling <- function(n)
{
  p <- runif(n, 0.001, 40)         # amostragem de uma Uniforme[0.001 , 40]
  d <- rep(1/39.999 , n)           # densidade da Uniforme
  return (list(points = p, density = d))
}

res1 <- sfbst.gamaexp(11, 12, bounds, const, initialPoint = init,
impSampFunction = sampling, custo = 10, alphacut = 0.75, rstar = 200)

message("Saída:")
print(res1$thetaStar)
print(res1$integral)
print(res1$sample.size)
print(res1$riscos)

#####################################################################
Saída:
[1] 1          (θ*, i.e., valor que maximiza a posteriori sob H0)
[1] 0.5515009                  (valor da evidência a favor de H0)
[1] 0        (quantidade de novas observações a serem realizadas)

       0        1         (riscos totais, i.e., incluído o custo)
147.0669 194.1644   (amostral da tomada de decisão imediatamente)
                              (e após 1 observação ser amostrada)
#####################################################################
```

<u>Exemplo 2</u>: Estimação da média de uma distribuição Normal (μ, $\sigma^2 = 2$).

Suponha-se que, após a realização de uma série de observações, a distribuição *a posteriori* para μ seja uma Normal (1, 0,2). A hipótese que se deseja testar é $H_0 : \mu = 1{,}1$ e os comandos abaixo produzem os resultados a seguir:

```
#####################################################################
# Limites da região do espaço paramétrico onde será buscado o ponto
# de máximo da posteriori sob H0
bounds <- list(lower = -10, upper = 10)
```

```
# Hipótese nula, escrita sob a forma de restrição de igualdade
const <- "1.1 - x1"

# Ponto inicial para o procedimento de otimização da posteriori
init <- 0.9

# Função de importância para o Importance Sampling
sampling <- function(n)
{
  p <- runif(n, -10, 10)          # amostragem de uma Uniforme[-10, 10]
  d <- rep(1/20 , n)              # densidade da Uniforme
  return (list(points = p, density = d))
}

res2 <- sfbst.normnorm(1, 0.2, 2, bounds, const, initialPoint = init,
impSampFunction = sampling, custo = 10, alphacut = 0.75, rstar = 200)

message("Saída:")
print(res2$thetaStar)
print(res2$integral)
print(res2$sample.size)
print(res2$riscos)

######################################################################
Saída:
[1] 1.1        (θ*, i.e., valor que maximiza a posteriori sob H0)
[1] 0.8285867                   (valor da evidência a favor de H0)
[1] 1        (quantidade de novas observações a serem realizadas)

       0        1        2 (riscos totais, i.e., incluído o custo)
137.1306 137.0906 151.3576         (amostral da tomada de decisão)
         (imediatamente e após 1 e 2 observações serem amostradas)
######################################################################
```

<u>Exemplo 3</u>: Independência em tabelas 2x2.

Considere-se a matriz abaixo (Pereira *et al*, 2008), referente a uma amostra de 224 estudantes submetidos a avaliação odontológica por dois dentistas,

		Dentista I		
		Aprovado	Reprovado	
Dentista II	Aprovado	62	41	103
	Reprovado	25	96	121
		87	137	224

e suponha-se que a distribuição *a priori* para o vetor (p_{11}, p_{12}, p_{21}, p_{22}) seja uma *Dirichlet* (1, 1, 1, 1). A hipótese que se deseja testar é $H_0 : p_{11}p_{22} = p_{12}p_{21}$ (independência/ausência de correlação entre avaliações dos dois dentistas) e os comandos abaixo produzem os resultados a seguir:

```
######################################################################
# Limites da região do espaço paramétrico onde será buscado o ponto
# de máximo da posteriori sob H0
bounds <- list(lower = c(0,0,0,0),upper = c(1,1,1,1))

# Hipótese nula, escrita sob a forma de restrições de igualdade
const <- c("(x1 * x4) - (x2 * x3)")
```

```
# Ponto inicial para o procedimento de otimização da posteriori
init <- c(0.25, 0.25, 0.25, 0.25)

# Função de importância para o Importance Sampling
sampling <- function(n)
{
  r1 <- runif(n, 0, 1)                    # gerando vetores no simplex
  r2 <- runif(n, 0, 1-r1)                 # r1 + r2 + r3 + r4 = 1
  r3 <- runif(n, 0, 1-(r1+r2))            # contido em [0,1]^4
  r4 <- 1 - (r1+r2+r3)
  p <- cbind(r1, r2, r3, r4)
  d <- 1 * (1/(1-r1)) * (1/(1-(r1+r2)))  # densidade dos vetores gerados
  return (list(points = p, density = d))
}

# Vetor de parâmetros da posteriori Dirichlet
alpha <- c(63,42,26,97)

# Restrição adicional de igualdade para definição do domínio
# (equivale a p11 + p12 + p21 + p22 = 1)
c1 <- "x1 + x2 + x3 + x4 - 1"

res3 <- sfbst.dirimult(alpha, bounds, const, equalDomainConstr = c1,
initialPoint = init, impSampFunction = sampling, custo = 10, alphacut =
0.75, rstar = 200)

message("Saída:")
print(res3$thetaStar)
print(res3$integral)
print(res3$sample.size)
print(res3$riscos)

######################################################################
Saída:
[1] 0.1785806 0.2812223 0.2098045 0.3303926  (θ*, i.e., vetor que
                                    maximiza a posteriori sob H0)
[1] 3.408019e-08               (valor da evidência a favor de H0)
[1] 0        (quantidade de novas observações a serem realizadas)

           0            1 (riscos totais, i.e., incluído o custo)
9.088050e-06 1.000001e+01         (amostral da tomada de decisão)
                (imediatamente e após 1 observação ser amostrada)
######################################################################
```

<u>Exemplo 4</u>: Independência entre Poissons.

Seja o vetor aleatório $(X,Y) \sim Holgate(\theta_1,\theta_2,\theta_3)$ amostrado conforme segue:

X	1	0	2	0	1	2	1	1	0	2	1	2	2	3	0
Y	2	0	0	0	0	0	0	0	1	2	2	1	1	2	1

A hipótese que se deseja testar é $\theta_3 = 0$ (independência entre X e Y) e o comando abaixo produz os resultados a seguir:

```
######################################################################

X <- c(1,0,2,0,1,2,1,1,0,2,1,2,2,3,0)
Y <- c(2,0,0,0,0,0,0,0,1,2,2,1,1,2,1)
```

```
res4 <- indep.holgate(X, Y, custo = 10, alphacut = 0.75, rstar = 200)

message("Saída:")
print(res4$thetaStar)
print(res4$integral)
print(res4$sample.size)
print(res4$riscos)

#####################################################################
Saída:
[1] 1.2 0.8 0.0                     (θ*, i.e., vetor que maximiza
                                        a verossimilhança sob H0)
[1] 0.6978041                   (valor da evidência a favor de H0)
[1] 3        (quantidade de novas observações a serem realizadas)

       0        1        2        3        4 (riscos totais, i.e.)
186.0811 164.7876 164.4018 158.5215 165.3717   (incluído o custo)
                                 (amostral, da tomada de decisão)
                               (imediatamente e após 1, 2, 3 e 4)
                                  (observações serem amostradas)
#####################################################################
```

Alguns comentários importantes devem ser tecidos acerca desses exemplos:

- Embora tenham sido destacados nesses exemplos apenas os valores `theta.star`, `integral`, `sample.size` e `riscos`, as variáveis de saída `res` (`res1`, `res2`, `res3` e `res4`) contêm outras estatísticas referentes aos processos de otimização e cálculo de evidência, todas descritas na documentação das rotinas, no Apêndice A.

- Tanto no Exemplo 1 quanto nos códigos-fonte do Apêndice A, foi utilizada a parametrização da distribuição Gama com parâmetros de escala α e de forma β, isto é, com densidade $f(x \mid \alpha, \beta) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$.

- Note-se que, no Exemplo 2, a quantidade apontada de novas observações a serem realizadas foi igual a um e, nos exemplos 1 e 3, foi igual a zero. Isso certamente se deve ao "chute" dos valores dos parâmetros *alphacut* e *rstar*. Acredita-se que um tomador de decisão que possuísse um maior conhecimento da natureza dos experimentos poderia anunciar valores de *alphacut* e *rstar* (ou mesmo dos parâmetros originais *a*, *b* e *c*) que conduzissem a resultados mais plausíveis para a quantidade de novas observações a serem realizadas.

- As funções *sampling* (distribuições de importância) também foi "chutada" e acredita-se igualmente que esse "chute" possa interferir nas quantidades apontadas de novas observações a serem tomadas. Novamente, acredita-se que, num problema real, alguém com conhecimento mais apropriado da natureza do experimento poderia formular uma distribuição de importância mais adequada do que as utilizadas nesses exemplos.

- Os (hiper)parâmetros da distribuição *a posteriori* são diretamente informados como argumentos das funções da biblioteca *sfbst.R*, o que é uma característica positiva dessas rotinas: enquanto a informação de tais (hiper)parâmetros é uma tarefa simples e rápida, a expressão de uma equação de densidade (ou mesmo do logaritmo do seu *kernel*) é uma tarefa mais demorada e facilmente vulnerável a enganos de digitação, dada a quantidade usual de variáveis, parcelas, termos e parênteses aninhados envolvidos.

Capítulo 7
Convergência

Um aspecto importante a ser verificado sobre o FBST e sua versão seqüencial é a convergência da seqüência de evidências à medida que o tamanho da amostra aumenta. Neste capítulo são exibidos alguns resultados relativos às distribuições conjugadas da família exponencial canônica, classe em que se enquadra parcela significativa dos problemas em que usualmente se recorre ao FBST.

7.1. Distribuição assintótica da seqüência de posterioris

De acordo com Bernardo e Smith (1994), sendo $x_1, x_2, x_3, \ldots$ observações de uma variável aleatória da família exponencial canônica, com *priori* canônica conjugada $P_0(\theta)$ e *posterioris* $P_1, P_2, P_3, \ldots$, onde $p_n(\theta) = \xi(\theta \mid x_1, x_2, \ldots, x_n)$, e definindo:

- m_n como o ponto de máximo da densidade p_n (ou, equivalentemente, de seu logaritmo $\ell_n(\cdot)$);
- $\Sigma_n = (-\ell''_n(m_n))^{-1}$, onde $\ell''_n(m_n)$ é a segunda derivada de $\ell_n(\cdot)$, avaliada no ponto m_n;

então a distribuição assintótica de P_n é $N_k(m_n; \Sigma_n)$, onde k é a dimensão do espaço paramétrico Θ.

Além disso, definindo $\phi_n = \Sigma_n^{-1/2}(P_n - m_n)$, Bernardo e Smith (1994) mostram que

$\phi_n \xrightarrow{\mathcal{D}} N_k(0_n; \mathbf{I}_n)$, onde:

- 0_n é o vetor de zeros de ordem *n* e
- $\mathbf{I}_n$ é a matriz identidade de ordem *n*.

7.2. Distribuição assintótica da seqüência de evidências

Dentre as diversas situações em que as distribuições conjugadas pertençam à família exponencial canônica, no caso particular em que $k = 1$ (ou seja, em que as *prioris/posterioris* sejam uniparamétricas), pode se examinar a questão da convergência da seqüência de evidências, conforme o desenvolvimento a seguir.

Inicialmente, tem-se, da seção anterior, que, para *n* suficientemente grande, $P_n \sim N(m_n; \Sigma_n)$, o que permite considerar que todos os testes uniparamétricos (em distribuições exponenciais canônicas com *prioris* canônicas conjugadas) sejam assintoticamente equivalentes a um teste para a média de uma distribuição Normal com varância conhecida.

Além disso, Fossaluza *et al* (2005) mostraram que para tal teste ($X \sim N(\theta, \sigma^2)$ e $H_0 : \theta = \theta_0$):

- $Ev(\Theta_0, P_n) \xrightarrow{\mathrm{D}} Uniforme[0,1]$, quando $\theta = \theta_0$ (H_0 verdadeira);
- $Ev(\Theta_0, P_n) \xrightarrow{q.c.} 0$, quando $\theta \neq \theta_0$ (H_0 falsa);

Assim, pode-se conjecturar que, em geral (para qualquer dimensão *k* do espaço paramétrico), para os testes envolvendo distribuições exponenciais canônicas com *prioris* canônicas conjugadas, esse também seja o comportamento assintótico da Evidência contra a hipótese nula, conforme verificado por Pereira *et al* (2008) para algumas situações particulares.

Pelas mesmas razões de intratabilidade apresentadas ao final da Seção 4.2, é inviável a verificação analítica dessa conjectura. Assim, recorreu-se a simulações para auxiliar a percepção dessas e outras características relacionadas à convergência (ou não) dos valores das medidas de evidência.

Para todos os casos relacionados na Seção 6.1, calculou-se a evidência *a priori* (i.e. obtida a partir da distribuição *a priori*, sem efetuar nenhuma observação) e conjuntos de evidências (obtidas por meio de simulações) para tamanhos de amostra variando de 1 a 10000. Para cada caso, foram realizadas duas séries distintas de simulações: uma com o mecanismo gerador da amostra parametrizado segundo a hipótese nula (ou seja, a amostra é gerada sob H_0 verdadeira) e outra com parâmetros suficientemente alterados (fazendo a amostra ser gerada sob H_0 falsa).

Nessas simulações, em diversos casos com tamanho de amostra muito alto, as evidências não puderam ser calculadas, sendo geradas mensagens de erro. Isso presumivelmente se deve ao fato de tais tamanhos de amostra produzirem valores muito altos para os parâmetros da *posteriori* e o *software* R não ser capaz de processar esses valores.

Os gráficos a seguir, produzidos a partir dessas simulações, endossam claramente a conjectura de que, quando H_0 é verdadeira, a distribuição das evidências converge para uma Uniforme[0,1] e que, quando H_0 é falsa, a seqüência de evidências converge para 0.

Note-se, porém, nas simulações sob H_0 falsa, que, embora na maioria das vezes a seqüência de evidências convirja decrescentemente para 0, na situação 3.2 (Figura 4), a seqüência cresce até atingir um valor máximo em n=5 e somente depois desse ponto começa a decrescer até chegar a zero. Além disso, nas situações 4.2 (Figura 5) e 5.2 (Figura 6), a seqüência de evidências apresenta um comportamento oscilatório, entre e 0,03 e 0,06 e entre 0,16 e 0,18, respectivamente. Isso sugere que, para esses casos, a convergência para zero seja mais vagarosa e que tais oscilações possam ser atribuídas à variabilidade das amostras simuladas.

Esses gráficos, porém, podem não ser suficientes para se concluir se o comportamento observado nessa situações é significativo ou até que ponto as características percebidas em cada uma das situações abordadas podem ser extrapoladas a qualquer problema envolvendo as distribuições em questão.

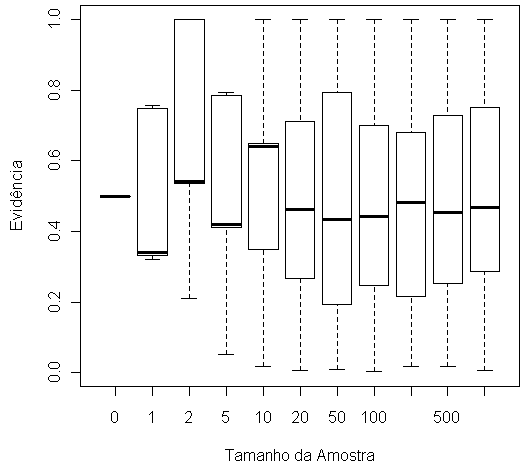
Caso 1: **Verossimilhança Binomial x *Priori*/*Posteriori* Beta**

Verossimilhança: $X \sim Binomial(n, p)$

Priori: $p \sim Beta(7,5)$

$H_0 : p = 0{,}5$

1.1: simulações sob H_0 verdadeira
(mecanismo gerador: $X \sim Binomial(n, 0{,}5)$)



1.2: simulações sob H_0 falsa
(mecanismo gerador: $X \sim Binomial(n, 0{,}58)$)

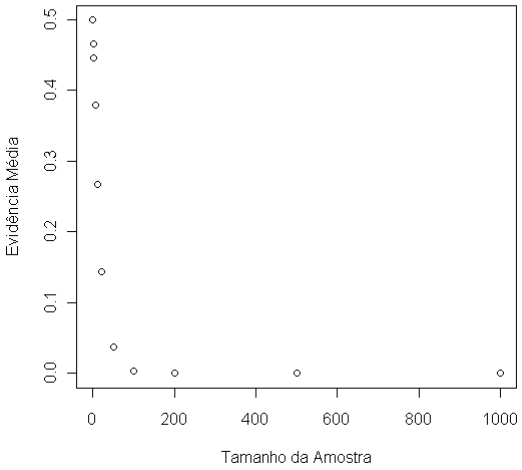


Figura 2 - Simulações para o caso Binomial x Beta

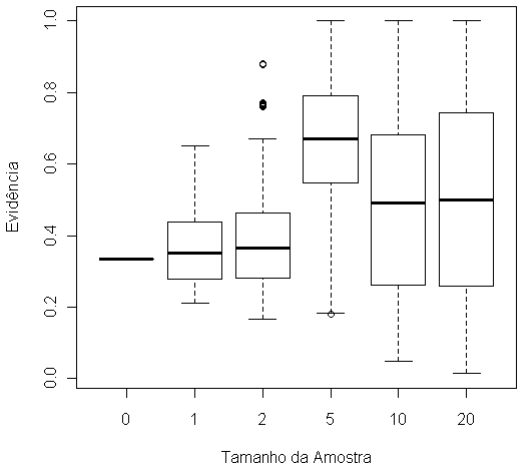
Caso 2: **Verossimilhança Poisson x *Priori*/*Posteriori* Gama**

Verossimilhança: $X \sim Poisson(\lambda)$

Priori: $\lambda \sim Gama(37,21)$

$H_0 : \lambda = 2$

2.1: simulações sob H_0 verdadeira
(mecanismo gerador: $X \sim Poisson(2)$)



2.2: simulações sob H_0 falsa
(mecanismo gerador: $X \sim Poisson(1{,}75)$)

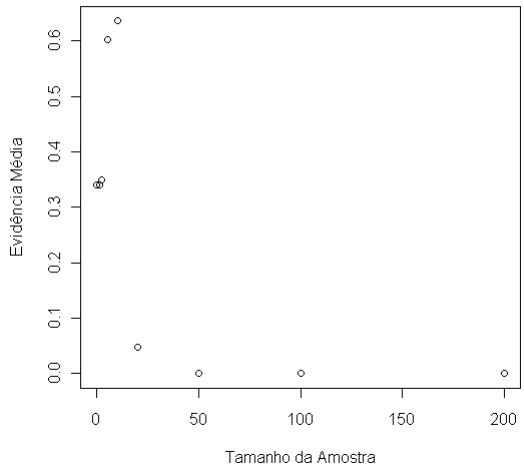


Figura 3 - Simulações para o caso Poisson x Gama

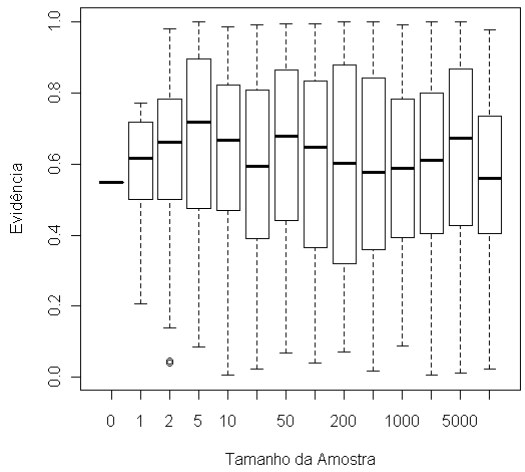
Caso 3: **Verossimilhança Exponencial x *Priori*/*Posteriori* Gama**

Verossimilhança: $X \sim Exponencial(\lambda)$

Priori: $\lambda \sim Gama(11,12)$

$H_0 : \lambda = 1$

3.1: simulações sob H_0 verdadeira
(mecanismo gerador: $X \sim Exponencial(1)$)



3.2: simulações sob H_0 falsa
(mecanismo gerador: $X \sim Exponencial(\lambda = 2,1)$)

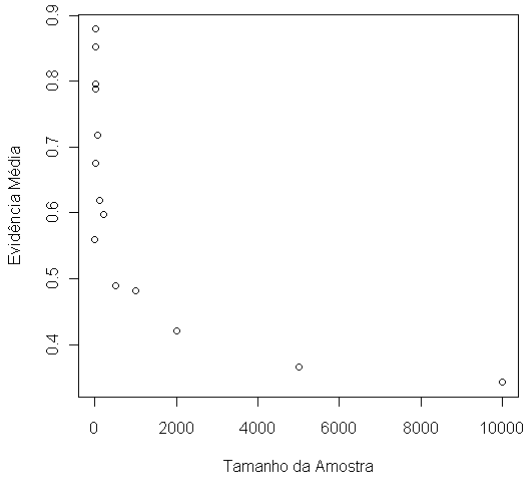


Figura 4 - Simulações para o caso Exponencial x Gama

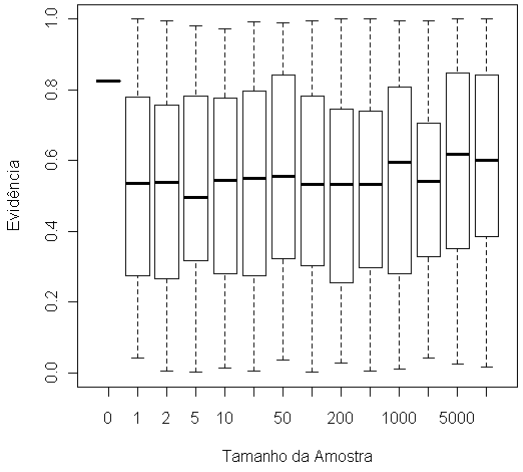
Caso 4: **Verossimilhança Normal (variância conhecida) x *Priori*/*Posteriori* Normal**

Verossimilhança: $X \sim N(\mu, \sigma^2 = 2)$

Priori: $\mu \sim N(1, s^2 = 0,2)$

$H_0 : \mu = 1,1$

4.1: simulações sob H_0 verdadeira
(mecanismo gerador: $X \sim N(1,1, \sigma^2 = 2)$)



4.2: simulações sob H_0 falsa
(mecanismo gerador: $X \sim N(2,5, \sigma^2 = 2)$)

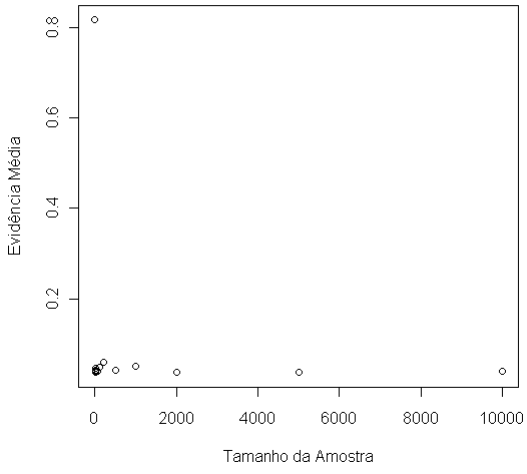


Figura 5 - Simulações para o caso Normal (variância conhecida) x Normal

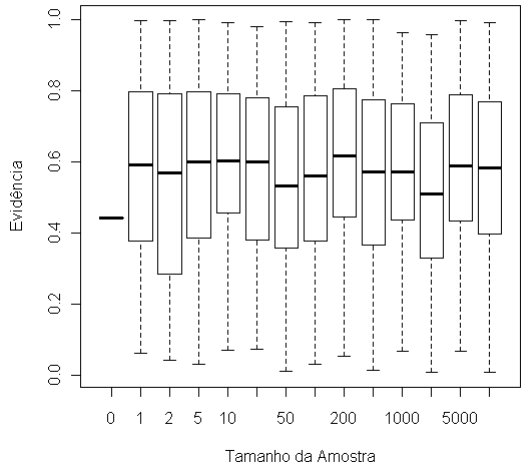
Caso 5: **Verossimilhança Normal (média conhecida) x *Priori*/*Posteriori* Gama**

Verossimilhança: $X \sim N\left(\mu = 0, \sigma^2 = \frac{1}{\tau}\right)$

Priori: $\tau \sim Gama(26,29)$

$H_0 : \tau = \sigma^2 = 1$

5.1: simulações sob H_0 verdadeira
(mecanismo gerador: $X \sim N(0,1)$)



5.2: simulações sob H_0 falsa
(mecanismo gerador: $X \sim N(0, \sigma^2 = 0{,}45)$)

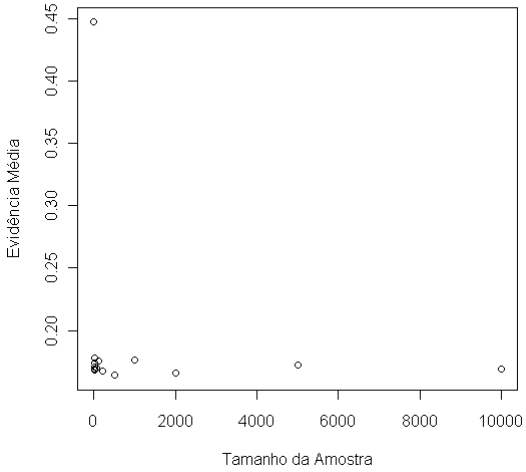


Figura 6 - Simulações para o caso Normal (média conhecida) x Gama

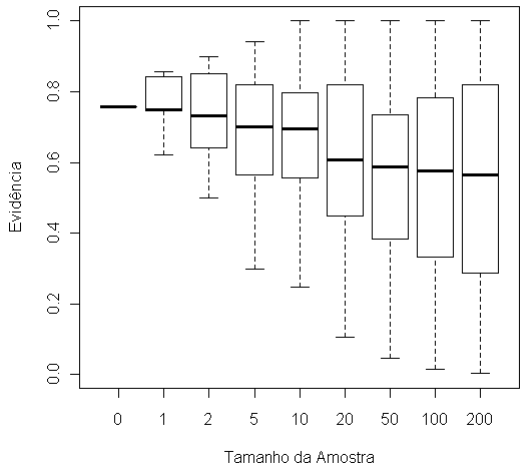
Caso 6: **Verossimilhança Multinomial x *Priori*/*Posteriori* Dirichlet**

Verossimilhança: $X \sim Multinomial(n, p_1, p_2, p_3, p_4)$

Priori: $p \sim Dirichlet(15,10,12,12)$

$H_0 : (p_1, p_2, p_3, p_4) = (0{,}25\ ,\ 0{,}25\ ,\ 0{,}25\ ,\ 0{,}25)$

6.1: simulações sob H_0 verdadeira
(mecanismo gerador:
$X \sim Multinomial(n, 0{,}25\ ,\ 0{,}25\ ,\ 0{,}25\ ,\ 0{,}25)$)



6.2: simulações sob H_0 falsa
(mecanismo gerador:
$X \sim Multinomial(n, 0{,}1\ ,\ 0{,}2\ ,\ 0{,}3\ ,\ 0{,}4)$)

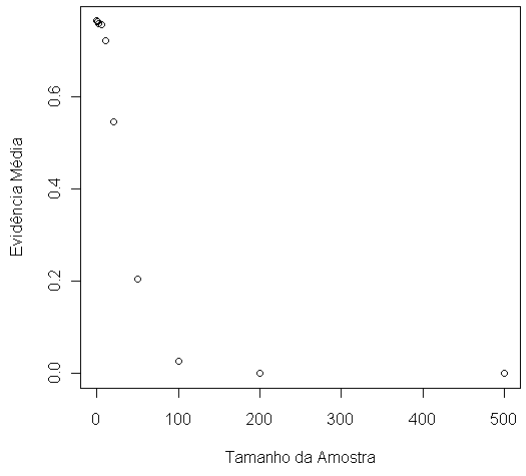


Figura 7 - Simulações para o caso Multinomial x Dirichlet

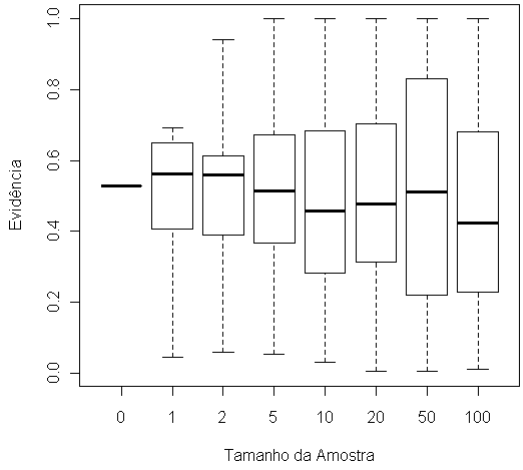
Caso 7: **Verossimilhança Normal (média e variância desconhecidas) x *Priori*/*Posteriori* Normal-Gama**

Verossimilhança: $X \sim N\left(\mu, \sigma^2 = \frac{1}{\tau}\right)$

Priori: $(\mu, \tau) \sim Normal\text{–}Gama(1,0{,}5,19,15)$

$H_0 : \mu = 0\,,\ \sigma^2 = \tau = 1$

7.1: simulações sob H_0 verdadeira (mecanismo gerador: $X \sim N(0,1)$)



7.2: simulações sob H_0 falsa (mecanismo gerador: $X \sim N(1{,}2\,,\ \sigma^2 = 2)$)

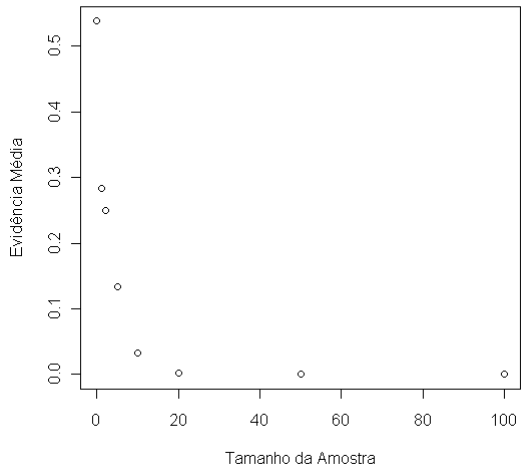


Figura 8 - Simulações para o caso Normal (média e variância desconhecidas) x Normal-Gama

Observação: nos casos 2, 3 e 5, foi utilizada a parametrização da distribuição Gama com parâmetros de escala α e de forma β, isto é, com densidade $f(x \mid \alpha, \beta) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$.

Capítulo 8
Considerações finais

Foram construídos, nesta tese, algoritmos e seus respectivos *softwares* (em linguagem R) para execução do FBST Seqüencial em situações em que as distribuições *priori/posteriori* e a verossimilhança são conjugadas e também para problemas específicos como o teste de independência entre Poissons, os testes para tabelas 2x2 e o problema de Behrens-Fisher. Um progresso colateral trazido por esse *software* é seu processamento inteiramente interno ao ambiente R, prescindindo da chamada a programas externos e conseqüentemente dispensando a necessidade de tê-los adequadamente instalados e configurados.

Em que pesem esse progresso computacional e o bom desempenho geral do *software* (na medida em que realiza eficientemente as tarefas para as quais foi concebido), esta tese é somente um passo inicial num tema em que ainda há muito a ser desbravado. Assim, várias considerações se fazem necessárias, tanto com relação ao alicerçamento e extensão da teoria desenvolvida nesta tese, quanto a respeito da execução do *software* apresentado.

O primeiro aspecto em que se pode investir em pesquisas futuras é a procura de um tratamento do SFBST nos moldes do teste seqüencial de Wald (1945), isto é, através da construção de funções *W*, *L* e *U* tais que o processo de decisão se assemelhe a:

$$\begin{cases} \text{Se } W < L\text{, rejeite } H_0 \\ \text{Se } L < W < U\text{, continue a amostragem} \\ \text{Se } W > U\text{, aceite } H_0 \end{cases} \quad \text{ou} \quad \begin{cases} \text{Se } W < L\text{, aceite } H_0 \\ \text{Se } L < W < U\text{, continue a amostragem} \\ \text{Se } W > U\text{, rejeite } H_0 \end{cases}$$

Também é uma área em aberto, podendo ensejar muitas pesquisas futuras, a questão, abordada na Seção 4.3.3, da obtenção de valores com interpretação e significado concretos, que possam substituir os parâmetros *a*, *b* e *c* relativos aos riscos de aceitação e rejeição da hipótese nula. Ainda nesse tópico, pode-se desenvolver estudos voltados ao parâmetro de corte α e a formas de determiná-lo por intermédio de níveis de significância ou conceitos similares.

Outro campo em que se pode empreender pesquisas significativas é a generalização dos algoritmos apresentados ao nesta tese a modelos hierárquicos e/ou a situações em que o custo amostral *C* não seja constante.

Com respeito às (potencialmente incontáveis) aplicações práticas do FBST Seqüencial, uma em que se pode empreender importantes desenvolvimentos é a Lei de Equilíbrio de Hardy-Weinberg (Pereira e Stern (1999), Montoya-Delgado *et al* (2001), Faria Jr. (2006)), de fundamental papel na área de genética populacional. Também pode ser bastante produtiva a aplicação do FBST Seqüencial a problemas de verificação de paternidade (Montoya-Delgado, 1998).

No que tange ao algoritmo propriamente dito, uma linha sugerida de trabalho é a sua integração com mecanismos de simulação de *posterioris* (como, por exemplo, o ABC), sobretudo nos casos em que as distribuições envolvidas não sejam conjugadas. Também se pode estudar e propor aperfeiçoamentos a aspectos tais como tempo de processamento e medidas de performance (qualidade ou precisão) dos resultados fornecidos.

Outra linha de pesquisa em que se pode investir é a comparação do FBST Seqüencial com outros testes seqüenciais (como por exemplo o teste seqüencial de Wald (1945)) com relação aos tamanhos esperados das seqüências de amostras até a decisão terminal ser atingida. Também poderá produzir resultados interessantes a comparação, por intermédio de riscos, custos ou outros indicadores, de tais tamanhos esperados de amostra com tamanhos fixos de amostra para o FBST padrão, não-seqüencial.

Por fim, também se apresenta como uma área aberta a pesquisas futuras a investigação mais acurada sobre convergência e distribuição assintótica das evidências, buscando, entre outros resultados, as confirmações formais dos comportamentos observados nos gráficos da Seção 7.2, especialmente para hipóteses nulas não pontuais.

Capítulo 9
Referências

9.1. Referências bibliográficas

Anderson (1999) Eric C. Anderson. Monte Carlo Methods and Importance Sampling. Em *Lecture Notes for Statistical Genectics*, volume 578C. Citado nas págs. 25 e 32.

Bernardo e Smith (1994) José M. Bernardo e Adrian F. M. Smith. *Bayesian Theory*. John Wiley & Sons, 1ª edição. Citado nas págs. 5 e 38.

Campos *et al* (2005) Pedro S. S. Campos, Maria Regina Madruga e Héliton R. Tavares. Teste de Significância Bayesiano no Problema de Behrens-Fisher. Em *XXVIII CNMAC- Congresso Nacional de Matemática Aplicada e Computacional*. Citado nas págs. 29 e 30.

DeGroot (1970) Morris DeGroot. *Optimal Statistical Decisions*, McGraw Hill. 1ª edição. Citado nas págs. 1, 2, 6 e 15

DeGroot (2004) Morris DeGroot. *Optimal Statistical Decisions*, Wiley-Interscience. 2ª edição. Citado nas págs. ii, iii, 1 e 18.

Faria Jr. (2006) Silvio R. Faria Jr. *Um ambiente computacional para um teste de significância bayesiano*. Dissertação de Mestrado, Instituto de Matemática e Estatística, Universidade de São Paulo, Brasil. Citado nas págs. 1, 21, 32, 44, 49 e 65.

Fink (1997) Daniel Fink. A Compendium of Conjugate Priors. Relatório Técnico, Department of Biology, Montana State University, Bozeman, USA. Citado na pág. 33.

Fossaluza *et al* (2005) Victor Fossaluza, José Adolfo de Almeida Schultz e Luiz Gustavo Esteves. Testes de hipóteses precisas: uma abordagem Bayesiana. Relatório de Pesquisa, Instituto de Matemática e Estatística, Universidade de São Paulo, Brasil. Citado nas pág. 39.

Holgate (1964) P. Holgate. Estimation for the bivariate Poisson distribution. *Biometrika*, 51: 241-245. Citado na pág. 25

Jeffreys (1961) Harold Jeffreys. *Theory of Probability*, Oxford University Press. 3ª edição. Citado na pág. 1.

Klarmann (1996) Patrícia C. Klarmann. *Planejamento Bayesiano de Ensaios Clínicos Seqüenciais.* Dissertação de Mestrado, Instituto de Matemática e Estatística, Universidade de São Paulo, Brasil. Citado nas págs. 2, 6, 9, 11, 15 e 16.

Lange (2004) Kenneth Lange. *Optimization*, Springer-Verlag. 1ª edição. Citado na pág. 32.

Madruga (2002) Maria Regina Madruga. *Teste de Significância: Uma proposta Genuinamente Bayesiana*. Tese de Doutorado, Instituto de Matemática e Estatística, Universidade de São Paulo, Brasil. Citado nas págs. 29 e 30.

Madruga *et al* (2001) Maria Regina Madruga, Luís Gustavo Esteves e Sergio Wechsler. On the Bayesianity of Pereira-Stern tests. *Test*, 10: 291-299. Citado nas págs. 1 e 5.

Madruga *et al* (2003) Maria Regina Madruga, Carlos Alberto B. Pereira e Julio Michael Stern. Bayesian evidence test for precise hypotheses. *Journal of Statistical Planning and Inference*, 117 (2): 185-198. Citado nas págs. 29 e 30.

Madsen *et al* (2004) K. Madsen, H. B. Nielsen e O. Tingleff. Optimization With Constraints. Relatório Técnico, Department of Mathematical Modelling, Techical University of Denmark, Lyngby, Denmark. Citado na pág. 32.

Marjoram *et al* (2003) Paul Marjoram, John Molitor, Vincent Plagnol e Simon Tavaré. Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences of the USA*, 100 (26): 15324-15328. Citado na pág. 21.

Montoya-Delgado (1998) Luis Eduardo Montoya-Delgado. *Probabilidade de Paternidade Uma Proposta Metodológica para Seu Cálculo.* Dissertação de Mestrado, Instituto de Matemática e Estatística, Universidade de São Paulo, Brasil. Citado na pág. 44.

Montoya-Delgado *et al* (2001) Luis Eduardo Montoya-Delgado, Telba Z. Irony, Carlos Alberto B. Pereira e Martin Whittle. Unconditional exact test for the Hardy-Weinberg Law: sample-space ordering using the Bayes Factor. *Genetics*, 158: 875-883. Citado na pág. 44.

Pereira e Stern (1999) Carlos Alberto B. Pereira e Julio Michael Stern. Evidence and credibility: full Bayesian significance test for precise hypotheses. *Entropy*, 1: 104-115. Citado nas págs. ii, iii, 1, 3 e 44.

Pereira *et al* (2008) Carlos Alberto B. Pereira, Julio Michael Stern e Sergio Wechsler. Can a Significance Test Be Genuinely Bayesian?, *Bayesian Analysis*, 3 (1): 79-100. Citado nas págs. 28, 35 e 39.

Stern (2003) Julio Michael Stern. Significance Tests, Belief Calculi, and Burden of Proof in Legal and Scientific Discourse. Laptec'03. *Frontiers in Artificial Intelligence and its Applications*, 101: 139-147. Citado na pág. 21.

Stern e Zacks (2002) Julio Michael Stern, Shelemyahu Zacks. Testing the Independence of Poisson Variates under the Holgate Bivariate Distribution: The Power of a New Evidence Test. *Statistical and Probability Letters*, 60: 313-320. Citado nas págs. 21 e 25.

Wald (1945) Abraham Wald. Sequential Tests of Statistical Hypotheses. *The Annals of Mathematical Statistics*, 16 (2): 117-186. Citado nas págs. 44 e 45.

9.2. Outras fontes consultadas

Goedman *et al* (2011) Rob Goedman, Gabor Grothendieck, Søren Højsgaard e Ayal Pinkus. Ryacas - an R interface to the yacas computer algebra system. http://cran.r-project.org/web/packages/Ryacas/Ryacas.pdf, 2011. Último acesso em 18/02/2012.

Irony e Pereira (2008) Carlos Alberto B. Pereira e Telba Z. Irony. Exact Tests for Equality of Two Proportions: Fisher v. Bayes. *Journal of Statistical Computation and Simulation*, 25: 93-114.

Kadane (2011) Joseph B. Kadane. *Principles of Uncertainty.* CRC Press. 1ª edição.

Kim e Cohen (1998) Seock-Ho Kim e Allan S. Cohen. On the Behrens-Fisher Problem: a Review. *Journal of Educational and Behavioural Statistics*, 23: 356-377.

Lauretto *et al* (2003) Marcelo Lauretto, Carlos Alberto B. Pereira, Julio Michael Stern e Shelemyahu Zacks. Full Bayesian significance test applied to multivariate normal structure models. *Brazilian Journal of Probability and Statistics*, 17: 147-168.

McNemar (1947) Quinn McNemar. Note on the Sampling Error of the Difference Between Correlated Proportion or Percentages. *Psychometrika*, 12(2): 153-157.

O'Hagan (1994) Anthony O'Hagan. *Kendall's Advanced Theory of Statistics, Vol. 2B: Bayesian Inference*, Edward Arnold Halsted Press. 1ª edição.

Parmigiani e Inoue (2009) Giovanni Parmigiani e Lurdes Inoue. *Decision Theory: Principles and Approaches*, John Wiley & Sons. 1ª edição.

Pereira (2011) Carlos Alberto B. Pereira. Full Bayesian Significance Test (FBST). Em Miodrag Lovric, organizador, *International Encyclopedia of Statistical Science Part 6*, páginas 551-554, primeira edição.

Pereira e Wechsler (1993) Carlos Alberto B. Pereira e Sergio Wechsler. On the concept of *P*-value. *Revista Brasileira de Probabilidade e Estatística*, 7: 159-177.

Rice (2010) Kenneth Rice. A Decision-Theoretic Formulation of Fisher's Approach to Testing. *The American Statistician*, 64(4): 345-349.

Stern (2007) Julio Michael Stern. Cognitive Constructivism, Eigen-Solutions, and Sharp Statistical Hypotheses. *Third Conference on the Foundations of Information Science. FIS2005*, 61: 1-23.

9.3. Sites

Algencan/Projeto Tango (*Trustable Algorithms for Nonlinear General Optimization*): http://www.ime.usp.br/~egbirgin/tango/. Último acesso em 18/02/2012.

R - Biblioteca Alabama (*Augmented Lagrangian Adaptive Barrier Minimization Algorithm*): http://cran.r-project.org/web/packages/alabama/. Último acesso em 18/02/2012.

R - Biblioteca Ryacas: http://cran.r-project.org/web/packages/Ryacas. Último acesso em 18/02/2012.

R - *Software*: http://www.r-project.org. Último acesso em 18/02/2012.

Yacas (*Yet Another Computer Algebra System*): http://yacas.sourceforge.net. Último acesso em 18/02/2012.

Apêndice A

Códigos-fonte desenvolvidos

A.1. Biblioteca yacas0.R

Observações:

1. Originalmente, o nome dessa biblioteca era *yacas.R*; a alteração para *yacas0.R* foi motivada pela prevenção de eventuais ambigüidades em razão de a biblioteca *Ryacas* possuir uma função também de nome `yacas`.

2. As funções `Yacas2Fortran`, `translate`, `hessian`, `is.linear`, `non.null` e `indices` não são utilizadas nas rotinas aqui apresentadas mas foram mantidas (e atualizadas, vide Seção 6.1) por fazerem parte do código-fonte original desenvolvido por Faria Jr. (2006) e por poderem vir a ser úteis em desenvolvimentos futuros do *software*.

3. Essa e as demais bibliotecas se encontram disponíveis para *download* na página http://www.chancedegol.com.br/sfbst/sfbst.htm .

```
# ================================================================
# Arquivo: yacas0.r
# ================================================================
# Autor: Silvio Rodrigues de Faria Junior
# Adaptação: Marcelo Leme de Arruda
# ================================================================
#
# ================================================================
# Módulo: Rotinas de Interface do R com os comandos das libraries
# Ryacas e alabama
#
# Modulo adaptado a partir das rotinas (criadas por Silvio RF Jr)
# de Interface do R com o Yacas para a geração do script de input
# do otimizador ALGENCAN.
#
# Os pontos onde foram feitas alterações ou adaptações à rotina
# original estão identificados com a expressão "(adaptMLA)".
#
# Para as rotinas deste módulo, foram preservadas integralmente
# as descrições e observações originais do autor (Silvio RF JR).
# Anotações acrescentadas ou modificadas estão identificadas com a
# expressão "(obsMLA)".
#
# Last update of any of the component of this module:
#
# May 27, 2011.
#
#
# ****************************************************************
# ****************************************************************

library(gdata)  # trim function
library(Ryacas) # funções do yacas diretamente acessíveis pelo R (adaptMLA)
library(alabama)# funções de maximização análogas às executadas
                # pelo Algencan  (adaptMLA)

# yacasInstall()   # (adaptMLA)

# ================================================================
# Observação: (obsMLA)
```

```
# O comando yacasInstall() só é necessário na primeira vez em que
# a library Ryacas for carregada.
# ================================================================
#
# Função: yacasR
#
# ================================================================
# Descrição
# Função de interface do R com o ambiente de álgebra
# simbólica YACAS (Yet Another Computer Algebra System)
# criado e mantido por Ayal Pinkus et al.
# URL: http://yacas.sourceforge.net
# Versão utilizada: 1.0.57
#
# ================================================================
# (obsMLA) Esse ambiente é emulado pela library Ryacas.
# ================================================================
# Uso
# yacasR (expr)
# ================================================================
# Argumentos
# expr: comando Yacas a ser avaliado em formato texto,
#       pode ser um comando único ou um vetor de comandos.
# ================================================================
# Retorno
# result: expressão Yacas avaliada em formato texto.
# ================================================================
# Observação
# A expressão Yacas passada como argumento não deve conter
# erros de sintaxe. A ocorrência de erros de sintaxe da
# da linguagem Yacas causa o travamento da sessão R.
# ================================================================
# Observações: (obsMLA)
# No código-fonte original, essa função tinha o nome yacas.
# A mudança para yacasR foi realizada em razão de a library
# Ryacas também possuir uma função com o nome Yacas.
#
# ================================================================
# Exemplos: (adaptMLA)
#
# > yacasR("D(x) sin(x)")
# [1] "cos(x)"
# > yacasR("D(x1, x2) x1^2 + Exp(x2)")
# [1] "Deriv(x1, x2, x1^2 + exp(x2))"
# > yacasR("D({x1, x2}) x1^2 + Exp(x2)")
# [1] "2 * x1" "exp(x2)"
# > yacasR("D(x1) D(x2) Sin(x1)*Cos(x2)")
# [1] "-(cos(x1) * sin(x2))"
# > yacasR("D(x) Exp(x*a) + b")
# [1] "a*Exp(x*a)"
#
# ****************************************************************
# ****************************************************************
yacasR <- function (expr) {
result <- NULL

if (is.character(expr)) {
  aux1 <- notation(expr, mode = "Yacas")
  aux2 <- yacas(aux1)                       # (adaptMLA)
  if (is.atomic(unlist(aux2)))              # (adaptMLA)
    {
```

```
    aux3 <- as.character(unlist(aux2)[1])   # (adaptMLA)
    }
  else                                      # (adaptMLA)
  aux3 <- as.character(unlist(aux2)$text)   # (adaptMLA)

  if (aux3[1] == "list")
    {
    result <- NULL                          # (adaptMLA)
    for (i in 1:length(aux3)-1)             # (adaptMLA)
      {                                     # (adaptMLA)
      result[i] <- aux3[i+1]                # (adaptMLA)
      }                                     # (adaptMLA)
    }                                       # (adaptMLA)
  else                                      # (adaptMLA)
  result <- as.character(aux2)              # (adaptMLA)
}
else
result <- NA
return (trim(result))
}
# ===================================================================
#
# Função: varList
#
# ===================================================================
# Descrição
# Extrai a lista de variáveis de uma expressão algébrica
# no formato da linguagem Yacas.
# ===================================================================
# Uso
# varList (expr)
# ===================================================================
# Argumentos
# expr: expressão algébrica Yacas a ser avaliada em formato string.
# ===================================================================
# Retorno
# Vetor de variáveis no formato string.
# ===================================================================
# Exemplos:
#
# > varList("x * y + z")
# [1] "x" "y" "z"
# > varList("x * Exp(y) + z")
# [1] "x" "y" "z"
# > varList("x * Exp(y) + Log(z)")
# [1] "x" "y" "z"
#
# *******************************************************************
# *******************************************************************
varList <- function(expr) {
vars <- yacasR(paste("VarList(",expr,")"))  # (adaptMLA)
return (vars)
}
# ===================================================================
#
# Função: translator
#
# ===================================================================
# Descrição
# Traduz uma expressão ou um vetor de expressões de uma
# linguagem para outra através de um dicionário fornecido.
```

```
# ====================================================================
# Uso
# translator (expr, dict)
# ====================================================================
# Argumentos
# expr: expressão algébrica ser traduzida em formato texto.
# dict: pairlist de strings para a tradução
# ====================================================================
# Retorno
# Expressão traduzida
# ====================================================================
# Exemplos:
#
# > dictionary <- pairlist(Exp = "exp", Log = "log")
# > translator("Exp(x) + Log(y)", dictionary)
# [1] "exp(x) + log(y)"
#
# ********************************************************************
# ********************************************************************
translator <- function (expr, dict) {
# Do nothing if dict is null
if (is.null(dict)) {
  on.exit(warning("Null dictionary"))
  return (expr)
  }
# Error handling
#print(is.what(expr))
if (!is.character(expr))
stop ("Parameter expr is not a text.")
if (!is.pairlist(dict))
stop ("Parameter dict is not a pairlist.")
# translation
newExpr <- expr
for (text in names(dict)) newExpr <- gsub(text, dict[text], newExpr)
return (newExpr)
}
# ====================================================================
#
# Função: Yacas2R
# R2Yacas
#
# ====================================================================
# Descrição
# Traduz uma expressão Yacas para R e vice-versa
# ====================================================================
# Uso
# Yacas2R(expr)
# R2Yacas(expr)
# ====================================================================
# Argumentos
# expr: expressão algébrica ser traduzida em formato texto.
# ====================================================================
# Retorno
# Expressão traduzida
# ====================================================================
# Exemplos:
#
# > Yacas2R("Ln(x) + Exp(y) + Cos(z)")
# [1] "log(x) + exp(y) + cos(z)"
# > R2Yacas("log(x) + exp(y) + cos(z)")
# [1] "Ln(x) + Exp(y) + Cos(z)"
```

```
#
# ********************************************************************
# ********************************************************************
Yacas2R <- function (expr) {
#Dictionary Yacas-R
Y2R <- pairlist (Exp = "exp",
Ln = "log",
Cos = "cos",
Sin = "sin",
Tan = "tan",
ArcSin = "asin",
ArcCos = "acos",
ArcTan = "atan",
Abs = "abs",
Sqrt = "sqrt",
Pi = "pi"
)
Rexp <- translator(expr, Y2R)
return (Rexp)
}
R2Yacas <- function (expr) {
R2Y <- pairlist (exp = "Exp",
log = "Ln",
cos = "Cos",
sin = "Sin",
tan = "Tan",
asin = "Asin",
acos = "ArcCos",
atan = "ArcTan",
abs = "Abs",
sqrt = "Sqrt",
pi = "Pi"
)
YExpr <- translator(expr, R2Y)
return (YExpr)
}
Yacas2Fortran <- function (expr) {
#Dictionary Yacas-R
Y2F <- pairlist (Exp = "EXP",
Ln = "LOG",
Cos = "COS",
Sin = "SIN",
Tan = "TAN",
ArcSin = "ASIN",
ArcCos = "ACOS",
ArcTan = "ATAN",
Abs = "ABS",
Sqrt = "SQRT",
Pi = "PI"
)
Fexp <- translator(expr, Y2F)
return (Fexp)
}
# ===================================================================
#
# Função: translate
#
# ===================================================================
# Descrição
# Traduz uma expressão algébrica de uma linguagem para outra
# ===================================================================
```

```
# Uso
# translate(expr, from = "R", to = "Yacas")
# ======================================================================
# Argumentos
# expr: expressão algébrica ser traduzida em formato texto.
# from: linguagem de origem da expressão.
# to: linguagem de destino.
# ======================================================================
# Retorno
# Expressão traduzida
# ======================================================================
# Exemplos:
#
# > translate("exp(x) + y + log(z)")
# [1] "Exp(x) + y + Ln(z)"
#
# **********************************************************************
# **********************************************************************
translate <- function (expr, from = "R", to = "Yacas") {
if (from == "R" && to == "Yacas")
return (R2Yacas(expr))
if (from == "Yacas" && to == "R")
return (Yacas2R(expr))
if (from == "Yacas" && to == "Fortran")
return (Yacas2Fortran(expr))
}
# ======================================================================
#
# Função: notation
#
# ======================================================================
# Descrição
# Transforma uma expressão com variáveis no formato
# x1, x2, ..., xn, com n número inteiro,
# para uma expressão com um vetor x com n posições
# ======================================================================
# Uso
# notation(expr, mode = "R", matr = FALSE)
# ======================================================================
# Argumentos
# expr: expressão algébrica a ser traduzida em formato texto.
# mode: linguagem de origem da expressão.
# matr: booleano indicador se traduz para formato
#       matricial na linguagem R.
# ======================================================================
# Retorno
# Expressão traduzida
# ======================================================================
# Exemplos:
#
# > notation("x1 + x2 + x3 + x4")
# [1] "x[1] + x[2] + x[3] + x[4]"
# > notation("x1 + x2 + x3 + x4", matr = TRUE)
# [1] "x[,1] + x[,2] + x[,3] + x[,4]"
#
# **********************************************************************
# **********************************************************************
notation <- function (expr, mode = "R", matr = FALSE) {
if (!is.character(expr))
stop ('Expression must be a string!')
res <- NULL
```

```
for (i in 1:length(expr)) {
  if (mode == "R" && !matr)
  res[i] <- gsub('x([0-9]+)', 'x[\\1]', Yacas2R(expr[i]))
  if (mode == "R" && matr)
  res[i] <- gsub('x([0-9]+)', 'x[,\\1]', Yacas2R(expr[i]))
  if (mode == "Yacas")
  res[i] <- gsub('[][]', '', R2Yacas(expr[i]))
  }
return (res)
}
# ====================================================================
#
# Função: gradient
#
# ====================================================================
# Descrição
# Avalia o gradiente de uma expressão algébrica do R
# ====================================================================
# Uso
# gradient(exprR, vars = NULL, order = 1, modeNotation = "R")
# ====================================================================
# Argumentos
# exprR: expressão algébrica em linguagem R.
# vars: variáveis que devem ser avaliadas no cálculo do gradiente.
# order: inteiro (1 ou 2) que indica a ordem da derivação
# modeNotation: linguagem que deve ser retornada o gradiente
#               avaliado.
# ====================================================================
# Retorno
# Lista do gradiente no formato string.
# ====================================================================
# Exemplos:
#
# > gradient("log(x) + exp(y)*cos(z)")
# D.x D.y D.z
# "1/x" "exp(y)*cos(z)" "-exp(y)*sin(z)"
# > gradient("log(x) + exp(y)*cos(z)", vars = c("x","z"))
# D.x D.z
# "1/x" "-exp(y)*sin(z)"
# > gradient("log(x) + exp(y)*cos(z)", vars = c("x","z"), order = 2)
# D.x D.z
# "(-1)/x^2" "-exp(y)*cos(z)"
# > gradient("x + y")
# D.x D.y
# "1" "1"
#
# ********************************************************************
# ********************************************************************
gradient <- function (exprR, vars = NULL,
                      order = 1, modeNotation = "R") {
exprY <- R2Yacas(exprR)
if (is.null(vars))
vars <- varList(exprY)
grad <- NULL
for (i in seq(length(vars))) {
  if (order == 1)
  g <- yacasR(paste("D(",vars[i],") ", exprY, sep=""))
  if (order == 2)
  g <- yacasR(paste("D(",vars[i],") D(",vars[i],") ", exprY, sep=""))
  grad[[i]] <- notation(trim(g), mode = modeNotation)
  }
```

```
names(grad) <- paste("D.", vars, sep="")
return (grad)
}
# ====================================================================
#
# Função: hessian
#
# ====================================================================
# Descrição
# Avalia a matriz hessiana de uma expressão algébrica R.
# ====================================================================
# Uso
# hessian(exprR, vars = NULL, modeNotation = "R")
# ====================================================================
# Argumentos
# exprR: expressão algébrica em linguagem R.
# vars: variáveis que devem ser avaliadas no cálculo das derivadas.
# modeNotation: linguagem que deve ser retornada o gradiente
#               avaliado.
# ====================================================================
# Retorno
# Matriz da hessiana em formato string.
# ====================================================================
# Exemplos:
#
# > hessian("x + log(y)*exp(z)")
# x y z
# x "0" "0" "0"
# y "0" "(-exp(z))/y^2" "exp(z)/y"
# z "0" "(y*exp(z))/y^2" "log(y)*exp(z)"
#
# > hessian("x + log(y)*exp(z)", vars = c("y","z"))
# y z
# y "(-exp(z))/y^2" "exp(z)/y"
# z "(y*exp(z))/y^2" "log(y)*exp(z)"
#
# > hessian("x+y")
#   x   y
# x "0" "0"
# y "0" "0"
#
# ********************************************************************
# ********************************************************************
hessian <- function (exprR, vars = NULL, modeNotation = "R") {
exprY <- R2Yacas(exprR)
grad <- NULL
if (!is.null(vars)) varsAux <- paste(vars, collapse = ",")
else {
  vars <- varList(exprY)
  varsAux <- paste(vars,collapse=",")
  }

l <- length(vars)                                       # (adaptMLA)
H <- array(0:0, c(l,l))                                 # (adaptMLA)

for (i in 1:l)                                          # (adaptMLA)
  for (j in 1:l)                                        # (adaptMLA)
    {                                                   # (adaptMLA)
    H[i,j] <- yacasR(paste("D(",vars[i],") D(",         # (adaptMLA)
              vars[j],") ", exprY, sep=""))             # (adaptMLA)
    }                                                   # (adaptMLA)
```

```
rownames(H) <- vars
colnames(H) <- vars
return (H)
}
# ====================================================================
#
# Função: is.linear
#
# ====================================================================
# Descrição
# Avalia se uma expressão algébrica é linear.
# ====================================================================
# Uso
# is.linear(expr, mode = "R")
# ====================================================================
# Argumentos
# expr: expressão algébrica.
# mode: linguagem que deve ser retornada o gradiente avaliado.
# ====================================================================
# Retorno
# 0 ou 1 para expressões algébricas em R
# .false. ou .true. para expressões algébricas em fortran
# ====================================================================
# Exemplos:
#
# > is.linear("x[1] + x[2]")
# [1] 1
# > is.linear("x[1] + sqrt(x[2])")
# [1] 0
#
# ********************************************************************
# ********************************************************************
is.linear <- function (expr, mode = "R") {
if (mode == "fortran") {
  fs = ".false."
  tr = ".true."
  } else {
  fs = 0
  tr = 1
  }
g <- gradient(expr)
t <- suppressWarnings(as.numeric(g))
if (is.element(NA, t)) return(fs)
else return(tr)
}
# ====================================================================
#
# Função: non.null
#
# ====================================================================
# Descrição
# Conta o número de elementos não nulos de uma matriz.
# ====================================================================
# Uso
# non.null(mat)
# ====================================================================
# Argumentos
# mat: matriz numérica.
# ====================================================================
# Retorno
```

```
# inteiro positivo
# ==========================================================
# Exemplos:
#
# > A <- cbind(c(1,2), c(3,4))
# > non.null(A)
# [1] 4
# > A <- cbind(c(0,2), c(3, 0))
# > non.null(A)
# [1] 2
# > A <- cbind(c(0,2), c(NA, 0))
# > non.null(A)
# [1] 2
#
# ***********************************************************
# ***********************************************************
non.null <- function (mat) {
A <- suppressWarnings(as.numeric(mat))
NAs <- sum(as.numeric(is.na(A)))
numbers <- sum(as.numeric(A != 0), na.rm = TRUE)
return(NAs + numbers)
}
# ==========================================================
#
# Função: indices
#
# ==========================================================
# Descrição
# Indica a ordem em que uma variável aparece numa expressão
# algébrica.
# ==========================================================
# Uso
# indices(expr, vars)
# ==========================================================
# Argumentos
# expr: expressão algébrica.
# vars: variáveis da expressão que devem ser indexadas.
# ==========================================================
# Retorno
# Vetor de inteiros que associa as variáveis indicadas à ordem
# em que elas ocorrem na expressão algébrica.
# ==========================================================
# Exemplos:
#
# > indices("a*x + b / log(y)", c("x", "y"))
# [1] 2 4
# > indices("a*x + b / log(y)", c("a"))
# [1] 1
# > indices("a*x + b / log(y)", c("a", "y"))
# [1] 1 4
#
# ***********************************************************
# ***********************************************************
indices <- function (expr, vars) {
varI <- varList(expr)
res <- NULL
for (v in vars)
res <- c(res, which(varI == v))
return (res)
}
###########################################################################
```

```
# ================================================================
#
# Função: otimiza
# Autor: Marcelo Leme de Arruda
# (utilizando alguns aspectos do código-fonte da função algencanR
# desenvolvida por Silvio Rodrigues de Faria Junior)
#
# ================================================================
# (obsMLA)
# Descrição
# Função de otimização que resolve problemas da forma
#
# minimizar f(x)
#
# sujeita a
#
# c_j(x) = 0, j em E,
# c_j(x) > 0, j em I,
#
# onde E é o conjunto dos índices das restricões de igualdade e
#      I é o conjunto dos índices das restricões de desigualdade, com
# x de dimensão n e m restricões.
#
# ================================================================
# (obsMLA)
# Observações:
# 1) Essa rotina substitui, por meio dos comandos da library
#    alabama, a chamada externa ao programa ALGENCAN.
# 2) IMPORTANTE: enquanto o Algencan trabalha as restrições
#    de desigualdade na forma c_j(x) <= 0, esta rotina, por
#    meio do comando auglag, da library alabama, manipula
#    tais restrições na forma c_j(x) > 0
#
# ================================================================
# (obsMLA)
# Uso
# otimiza (f, vars, init, cons,
#          maximize = FALSE,
#          lam0 = 10
#          sig0 = 100
#          eps = 1e-06
#          itmax = 100
#          trace - TRUE
#          method = "BFGS"
#          NMinit = FALSE
#          i.scale = 1
#          e.scale = 1
#           ...)
#
# ================================================================
# (obsMLA)
# Argumentos
# f: função algébrica a ser otimizada.
# vars: vetor de variáveis do problema.
# init: ponto inicial da otimização.
# cons: estrutura de dados que contém as expressões algébricas
#       das restricões aplicadas à função a ser minimizada e
#       os indicadores do tipo de restrição (identidades ou
#       desigualdades).
# maximize: booleano que indica se é para procurar o máximo da
#           função.
```

```
#
# Opcões do comando auglag
# ------------------------
# lam0
# sig0
# eps
# itmax
# trace
# method
# NMinit
# i.scale
# e.scale
#
# A descrição dessas variáveis pode ser encontrada na página
# http://finzi.psych.upenn.edu/R/library/alabama/html/auglag.html
#
# ================================================================
# (obsMLA)
# Retorno
#
# Lista com informações sobre a otimização realizada:
#
# par
# value
# counts
# convergence
# message
# outer.iterations
# lambda
# sigma
# gradient
# hessian
# ineq
# equal
# kkt1
# kkt2
#
# "par" é um vetor com o ponto de ótimo encontrado
# "value" é o valor da f(x) no ponto de mínimo
#        (ou de -f(x) no ponto de máximo de f)
#
# A descrição das demais informações também pode ser encontrada na página
# http://finzi.psych.upenn.edu/R/library/alabama/html/auglag.html
#
# ================================================================
# (obsMLA)
# Exemplos:
#
# > f <- "x2"
# > vars <- c("x1","x2")
# > cons <- list()
# > cons[['expression']] <- c("- x1^2 - 1 + x2", "- 2 + x1 + x2")
# > cons[['equality']] <- c(FALSE, FALSE)
# > init <- c(0,0)
# >
# > x <- otimiza(f, vars, init, cons, maximize = TRUE)
# > print(x$par)
# [1] 0.6180339 1.3819660
#
# > f <- "x2^(6.5) * exp(-0.5 *(x2 * (4 + 11*(x1 - 0.9)^2)))"
# > vars <- c("x1","x2")
```

```
# > cons <- list()
# > cons[['expression']] <- c("x1 - 1.1")
# > cons[['equality']] <- c(TRUE)
# > init <- c(1.1, 1)
# >
# > x <- otimiza(f, vars, init, cons, maximize = TRUE)
# > print(x$par)
# [1] 1.100000 2.927928
#
# > f <- "x1"
# > vars <- c("x1","x2")
# > cons <- list()
# > cons[['expression']] <- c("-(x1^2 + x2^2 - 1)")
# > cons[['equality']] <- c(FALSE)
# > init <- c(1.3, 0.1)
# >
# > x <- otimiza(f, vars, init, cons, maximize = TRUE)
# > print(x$par)
# [1] 1.000000e+00 2.596731e-17
#
# ******************************************************************
# ******************************************************************
otimiza <- function (f, vars, init, cons,
                       maximize = FALSE,
                       lam0 = 10,
                       sig0 = 100,
                       eps = 1e-07,
                       itmax = 50,
                       trace = FALSE,
                       method = "BFGS",
                       NMinit = FALSE,
                       i.scale = 1,
                       e.scale = 1,
                       ...
                       ) {


control.outer <- list(lam0 = lam0, sig0 = sig0, eps = eps, itmax = itmax,
                      method = method, trace = trace, NMinit = NMinit,
                      i.scale = i.scale, e.scale = e.scale)    # (adaptMLA)


# (obsMLA)
# Preparando (colocando em linguagem apropriada) a função fn a ser
# minimizada
# ================================================================

ftmp <- notation(f)                                            # (adaptMLA)
if (maximize)                                                  # (adaptMLA)
  { ftmp <- paste("-(",ftmp,")")                               # (adaptMLA)
  }                                                            # (adaptMLA)

script <- "fn0 <- function(x) {\n"                             # (adaptMLA)
script <- c(script, paste(ftmp), "}\n\n")                      # (adaptMLA)

# (obsMLA)
# Preparando (colocando em linguagem apropriada) o vetor gr, o
# gradiente de fn
# ================================================================

gtmp <- gradient(ftmp,vars)                                    # (adaptMLA)
```

```
script <- c(script,"gr0 <- function(x) {\n")                    # (adaptMLA)
script <- c(script,"g <- rep(NA, ",length(gtmp),") \n")         # (adaptMLA)
for (i in 1:length(gtmp))                                       # (adaptMLA)
   {                                                            # (adaptMLA)
   script <- c(script, "g[",i,"] <- ", paste(gtmp[i]),"\n")     # (adaptMLA)
   }                                                            # (adaptMLA)
script <- c(script,"g } \n\n")                                  # (adaptMLA)

# (obsMLA)
# Preparando (colocando em linguagem apropriada) a matriz heq
# de restrições de igualdade
# ================================================================

constmp <- list()                                               # (adaptMLA)
constmp$expression <- notation(cons$expression)                 # (adaptMLA)
constmp$equality <- cons$equality                               # (adaptMLA)
ncons <- length(constmp$equality)                               # (adaptMLA)
numeq <- 0                                                      # (adaptMLA)
for (i in 1:ncons)                                              # (adaptMLA)
if (constmp$equality[i])                                        # (adaptMLA)
  { numeq <- numeq + 1                                          # (adaptMLA)
  }                                                             # (adaptMLA)

if (numeq > 0)                                                  # (adaptMLA)
  {                                                             # (adaptMLA)
  script <- c(script,"heq0 <- function(x) {\n")                 # (adaptMLA)
  script <- c(script,"h <- rep(NA, ",numeq,") \n")              # (adaptMLA)
  ntmp <- 0                                                     # (adaptMLA)
  for (i in 1:ncons)                                            # (adaptMLA)
     if (constmp$equality[i])                                   # (adaptMLA)
       {                                                        # (adaptMLA)
       ntmp <- ntmp + 1                                         # (adaptMLA)
       script <- c(script, "h[",ntmp,"] <- ",                   # (adaptMLA)
                   paste(constmp$expression[i]),"\n")           # (adaptMLA)
       }                                                        # (adaptMLA)
  script <- c(script,"h } \n\n")                                # (adaptMLA)
}                                                               # (adaptMLA)

# (obsMLA)
# Preparando (colocando em linguagem apropriada) a matriz heq.jac,
# jacobiana das restrições de igualdade
# ================================================================

nvars <- length(vars)                                           # (adaptMLA)

if (numeq > 0)                                                  # (adaptMLA)
  {                                                             # (adaptMLA)
  script <- c(script,"heqjac0 <- function(x) {\n")              # (adaptMLA)
  script <- c(script,"j <- matrix(NA ,",numeq,",",nvars,") \n") #(adaptMLA)
  ntmp <- 0                                                     # (adaptMLA)
  for (i in 1:ncons)                                            # (adaptMLA)
     if (constmp$equality[i])                                   # (adaptMLA)
       {                                                        # (adaptMLA)
       ntmp <- ntmp + 1                                         # (adaptMLA)
       script <- c(script,"j[",ntmp,", ] <- c(")                # (adaptMLA)
       for (k in 1:nvars)                                       # (adaptMLA)
          {                                                     # (adaptMLA)
          d <- yacasR(paste("D(",vars[k],")",
               cons$expression[i]))                             # (adaptMLA)
          dnot <- notation(d)                                   # (adaptMLA)
          script <- c(script,paste(dnot))                       # (adaptMLA)
```

```
          if (k < nvars)                                          # (adaptMLA)
            script <- c(script,", ")                              # (adaptMLA)
          }                                                       # (adaptMLA)
          script <- c(script,")\n")                               # (adaptMLA)
       }                                                          # (adaptMLA)
  script <- c(script,"j } \n\n")                                  # (adaptMLA)
}                                                                 # (adaptMLA)

# (obsMLA)
# Preparando (colocando em linguagem apropriada) a matriz hin de
# restrições de desigualdade
# ================================================================

ncons <- length(constmp$equality)                                 # (adaptMLA)
numineq <- ncons - numeq                                          # (adaptMLA)

if (numineq > 0)                                                  # (adaptMLA)
  {                                                               # (adaptMLA)
  script <- c(script,"hin0 <- function(x) {\n")                   # (adaptMLA)
  script <- c(script,"h <- rep(NA, ",numineq,") \n")              # (adaptMLA)
  ntmp <- 0                                                       # (adaptMLA)
  for (i in 1:ncons)                                              # (adaptMLA)
     if (!(constmp$equality[i]))                                  # (adaptMLA)
       {                                                          # (adaptMLA)
       ntmp <- ntmp + 1                                           # (adaptMLA)
       script <- c(script, "h[",ntmp,"] <- ",                     # (adaptMLA)
                   paste(constmp$expression[i]),"\n")             # (adaptMLA)
       }                                                          # (adaptMLA)
  script <- c(script,"h } \n\n")                                  # (adaptMLA)
}                                                                 # (adaptMLA)

# (obsMLA)
# Preparando (colocando em linguagem apropriada) a matriz hin.jac,
# jacobiana das restrições de desigualdade
# ================================================================

if (numineq > 0)                                                  # (adaptMLA)
  {                                                               # (adaptMLA)
  script <- c(script,"hinjac0 <- function(x) {\n")                # (adaptMLA)
  script <- c(script,"j <- matrix(NA ,",numineq,",",nvars,
                    ") \n")                                       # (adaptMLA)
  ntmp <- 0                                                       # (adaptMLA)
  for (i in 1:ncons)                                              # (adaptMLA)
     if (!(constmp$equality[i]))                                  # (adaptMLA)
       {                                                          # (adaptMLA)
       ntmp <- ntmp + 1                                           # (adaptMLA)
       script <- c(script,"j[",ntmp,", ] <- c(")                  # (adaptMLA)
       for (k in 1:nvars)                                         # (adaptMLA)
          {                                                       # (adaptMLA)
          d <- yacasR(paste("D(",vars[k],")",
               cons$expression[i]))                               # (adaptMLA)
          dnot <- notation(d)                                     # (adaptMLA)
          script <- c(script,paste(dnot))                         # (adaptMLA)
          if (k < nvars)                                          # (adaptMLA)
            script <- c(script,", ")                              # (adaptMLA)
          }                                                       # (adaptMLA)
          script <- c(script,")\n")                               # (adaptMLA)
       }                                                          # (adaptMLA)
  script <- c(script,"j } \n\n")                                  # (adaptMLA)
}                                                                 # (adaptMLA)
```

```
# (obsMLA)
# Executando o comando de otimização
# ================================================================

cat(script, file = 'temp.r')                                        # (adaptMLA)
source("temp.r")                                                    # (adaptMLA)
fn <- fn0                                                           # (adaptMLA)
gr <- gr0                                                           # (adaptMLA)
if (numeq > 0)                                                      # (adaptMLA)
  { heq <- heq0                                                     # (adaptMLA)
    heq.jac <- heqjac0                                              # (adaptMLA)
  }                                                                 # (adaptMLA)
if (numineq > 0)                                                    # (adaptMLA)
  { hin <- hin0                                                     # (adaptMLA)
    hin.jac <- hinjac0                                              # (adaptMLA)
  }                                                                 # (adaptMLA)

if (numeq > 0)                                                      # (adaptMLA)
  if (numineq > 0)                                                  # (adaptMLA)
    result <- auglag(par=init,fn=fn,gr=gr,hin=hin,
                     hin.jac=hin.jac,heq=heq,heq.jac=heq.jac,
                     control.outer=control.outer)                   # (adaptMLA)

if (numeq > 0)                                                      # (adaptMLA)
  if (numineq == 0)                                                 # (adaptMLA)
    result <- auglag(par=init,fn=fn,gr=gr,heq=heq,heq.jac=heq.jac,
                     control.outer=control.outer)                   # (adaptMLA)

if (numeq == 0)                                                     # (adaptMLA)
  if (numineq > 0)                                                  # (adaptMLA)
    result <- auglag(par=init,fn=fn,gr=gr,hin=hin, hin.jac=hin.jac,
                     control.outer=control.outer)                   # (adaptMLA)

if (numeq == 0)                                                     # (adaptMLA)
  if (numineq == 0)                                                 # (adaptMLA)
    result <- c("ERRO: NÃO HÁ RESTRIÇÕES")                          # (adaptMLA)

result

}
#########################################################################
```

A.2. Biblioteca fbst.R

Observações:

1. A função `criticlevelEVA` não é utilizada nas rotinas aqui apresentadas mas foi mantida (e atualizada, vide Seção 6.1) por fazer parte do código-fonte original desenvolvido por Faria Jr. (2006) e por poder vir a ser útil em desenvolvimentos futuros do *software*.

2. Essa e as demais bibliotecas se encontram disponíveis para *download* na página http://www.chancedegol.com.br/sfbst/sfbst.htm .

```
# ====================================================================
# Arquivo: fbst.r
# ====================================================================
# Autor: Silvio Rodrigues de Faria Junior
# Adaptação: Marcelo Leme de Arruda
# ====================================================================
#
# ====================================================================
# Módulo: Rotinas para execução do teste FBST (Full Bayesian
# Significance Test).
#
# Observações:
# 1) Os pontos onde foram feitas alterações ou adaptações à
# rotina original estão identificados com a expressão "(adaptMLA)".
#
# 2) Para as rotinas deste módulo, foram preservadas integralmente
# as descrições e observações originais do autor (Silvio RF JR).
# Anotações acrescentadas ou modificadas estão identificadas com a
# expressão "(obsMLA)".
#
# Last update of any of the component of this module:
#
# June 21, 2011.
#
#
#########################################################################
source(“yacas0.r")                # (obs MLA) Interface do R com os
                                  # emuladores/substitutos do Yacas
                                  # e do ALGENCAN
# Validacao de Expressoes Algebricas
# Precisa ser implementado
is.validExpr <- function (expr) {
return (TRUE)
}
# ====================================================================
#
# Função: criticLevelEVA
#
# ====================================================================
# Descrição
# ---------
# Calcula o valor crítico da evidência CONTRÁRIA à hipótese nula #(obsMLA)
#
# ====================================================================
# Uso
# ---
# criticLevelEVA (dfTheta, dfNH, alpha)
#
# ====================================================================
# Argumentos
```

```
# ----------
# dfTheta: no. de graus de liberdade do espaço paramétrico.
# dfNH: no. de graus de liberdade da hipótese nula.
# alpha: nivel de confiança.
#
# ===================================================================
# Exemplos:
# > criticLevelEVA(3, 4, 0.2) # não retorna nada
# > criticLevelEVA(4, 2, 0.2)
# [1] 0.4781124
# > criticLevelEVA(4, 2, 0.9)
# [1] 0.005175536
# > criticLevelEVA(4, 2, 0.01)
# [1] 0.9439483
#
# *******************************************************************
# *******************************************************************
criticLevelEVA <- function (dfTheta, dfNH, alpha) {
if (dfTheta - dfNH > 0)
return(pchisq(qchisq(1 - alpha, dfTheta - dfNH), dfTheta))
}
# ===================================================================
#
# Função: evidence
#
# ===================================================================
# Descrição
# ---------
# Calcula a evidência contra a hipótese nula do FBST via
# Monte Carlo Importance Sample. O usuário deve definir a
# função de amostragem.
# ===================================================================
# Uso
# ---
# evidence (f,
#           thetaStar,
#           thetaMax = NULL,
#           impSampFunctn,
#           indicatorMethod = FALSE,
#           error = 0.01,
#           gamma = 0.99,
#           nDiv = 10,
#           step = 100,
#           nMax = 5e5,
#           nInit = 10000,
#           echoIt = FALSE)
# ===================================================================
# Argumentos
# ----------
# f: posteriori
# thetaStar: ponto de máximo sobre a Hipótese Nula.
# thetaMax: ponto de máximo irrestrito da posteriori.
# impSampFunctn: função de amostragem para o Monte Carlo Importance
#                Sampling
# indicatorMethod: booleano que indica se o calculo da evidencia
#                  será feito pelo método da variável indicadora.
# error: erro relativo da integração.
# gamma:
# nDiv:
# step: número de pontos amostrados a cada passo.
# nMax: número máximo de pontos amostrados para o processo todo.
```

```
# nInit: número de pontos amostrados no início do processo.
# echoIt: booleano que indica a impressão ou não de cada passo de
#         processo.
# ================================================================
# Retorno
# -------
# integral: evidência contra a hipótese nula para a posteriori avaliada.
# error: medida do erro cometido no cálculo da Evidência
# nPoints:
# thetaStar: ponto de máximo sobre a Hipótese Nula
# fStar: valor da posteriori no ponto thetaStar
# thetaMax: ponto de máximo irrestrito da posteriori.
# fMax: valor da posteriori no ponto thetaMax
# wLevel:
# wCount:
#
# ================================================================
# ****************************************************************
# ****************************************************************
evidence <- function (f,
                      thetaStar,
                      thetaMax = NULL,
                      impSampFunctn,
                      indicatorMethod = FALSE,
                      error = 0.01,
                      gamma = 0.99,
                      nDiv = 10,
                      step = 10000,
                      nMax = 1e6,
                      nInit = 10000,
                      echoIt = FALSE) {
if (!is.function(impSampFunctn))
  stop("A geradora do importance sampling nao estah definida")

fStar <- f(matrix(thetaStar, nrow = 1))

if (!is.null(thetaMax)) {
  fMax <- f(matrix(thetaMax, nrow = 1))
  if (fMax > 0)
  div <- seq(0, fMax, length.out = nDiv + 1)
else {
  div <- seq(-1770, fMax, length.out = nDiv + 1)
  }
}

if (indicatorMethod) {

  sample <- impSampFunctn(nInit)
  fPoints <- f(sample$points)
  W <- fPoints
  ZStar <- as.numeric(W > fStar)

  integral = mean(ZStar)
  estimatedError = sd(ZStar) / sqrt(length(ZStar) - 1)

  iter <- 1
  while(1) {
    sample <- impSampFunctn(step)

    fPoints <- f(sample$points)
    W <- c(W, fPoints)
```

```
    ZStar <- as.numeric(W > fStar)

    integral = mean(ZStar)
    estimatedError = sd(ZStar) / sqrt(length(ZStar) - 1)

    if (echoIt)
      message('Iter: ', iter, ' EVA: ', round(integral, 4),
              ' Error: ', round(estimatedError, 4),
              " points:", length(W))
    iter <- iter + 1

    if (!is.nan(estimatedError))
      if ((estimatedError > 0 && estimatedError < error))
        break
    if (length(W) > nMax)
      break
  }

} else {

  sample <- impSampFunctn(nInit)
  fPoints <- f(sample$points)

  W <- fPoints
  Z <- fPoints / sample$density

  ZStar <- 0*Z
  ZStar[W > as.numeric(fStar)] <- Z[W > as.numeric(fStar)]     # (adaptMLA)
  ZStarC <- 0*Z
  ZStarC[W <= as.numeric(fStar)] <- Z[W <= as.numeric(fStar)]  # (adaptMLA)
  estimatedError <- error + 1

  iter <- 1
  while(1) {
    sample <- impSampFunctn(step)

    fPoints <- f(sample$points)
    W <- c(W, fPoints)

    Z <- c(Z, fPoints / sample$density)

    ZStar <- 0*Z
    ZStar[W > as.numeric(fStar)] <- Z[W > as.numeric(fStar)]   # (adaptMLA)
    ZStarC <- 0*Z
    ZStarC[W <= as.numeric(fStar)] <- Z[W <= as.numeric(fStar)] #(adaptMLA)

    mi = mean(Z)
    miStar = mean(ZStar)
    csi = sd(Z)/mi
    csiStar = sd(ZStar)/mi
    csiStarC = sd(ZStarC)/mi

    etha = miStar/mi
    integral = etha

    A = (csiStar * (1-etha))
    B = (csiStarC * etha)
    C = (etha * (1 - etha))

    m = length(Z)
```

```
    estimatedError = sqrt((qchisq(gamma, 1) / m) * (A^2 + B^2 + 2*C^2))

    if (echoIt)
      message('Iter: ', iter, ' EVA: ', round(etha, 4),
              ' Error: ', round(estimatedError, 4), ' points:', m)
    iter <- iter + 1

    if ((estimatedError < error && !is.nan(estimatedError)))
      break
    if (length(Z) > nMax)
      break
  }

}
res <- list(integral = integral,
            error = estimatedError,
            nPoints = length(W),
            thetaStar = thetaStar,
            fStar = fStar)

if (!is.null(thetaMax)) {

  ncountW <- length(W);
  stepW <- floor(ncountW/nDiv);
  sortedW <- sort(W, decreasing=TRUE);
  div <- 1:nDiv
  for (i in 0:(nDiv-1)) div[nDiv-i] <- sortedW[1+i*stepW]
  div[nDiv] <- fMax
  resDiv <- NULL
  for (i in 1:nDiv)
    resDiv[i] <- sum(as.numeric(W <= div[i]))
  for (i in nDiv:2) {
    resDiv[i] <- resDiv[i] - resDiv[i-1]
    names(resDiv[i]) <- div[i]
  }

  res[['thetaMax']] <- thetaMax
  res[['fMax']] <- fMax
  res[['wLevel']] <- div
  res[['wCount']] <- resDiv
}

  return (res)
}
# ====================================================================
#
# Função: fbst
#
# ====================================================================
# Descrição
# ---------
# Implementação do FBST (Full Bayesian Significance Test)
# com a utilização da rotina otimiza (adaptMLA)
# Para o cálculo do valor de evidência do FBST, são necessários
# dois passos fundamentais:
# - O primeiro consiste na busca do ponto de máxima
# verossimilhança sobre a hipótese nula (H0), denominado theta*.
# - O segundo é a avaliação da integral da verossimilhança
# sobre a região dos pontos do espaço paramétrico que possuem
# valores na imagem superiores ao valor da imagem no ponto
```

```
# theta*.
#
#
# ==================================================================
# Uso
# ---
# fbst (optimizationExpr,
#       domainBounds,
#       equalNHConstr,
#       inequalNHConstr = NULL,
#       equalDomainConstr = NULL,
#       inequalDomainConstr = NULL,
#       initialPointNH,
#       integrationExpr = paste("exp(", optimizationExpr,")"),
#       impSampFunction,
#       indicatorMethod = FALSE,
#       evError = 0.01,
#       evGamma = 0.99,
#       evNDiv = 10,
#       evStep = 1000,
#       evNMax = 5e5,
#       evNInit = 9000,
#       evEchoIt = FALSE,
#       thetaMax = NULL,
#       ...)
# ==================================================================
# Argumentos
# ----------
# optimizationExpr: expressão da posteriori a ser otimizada
#                   é recomendável utilizar o log da posteriori.
# domainBounds: lista de lower e upper bounds da posteriori a ser
#               avaliada. Em caso de limitantes infinitos, definir
#               um valor suficientemente grande, mas limitado para
#               evitar problemas com o otimizador.
# equalNHConstr: expressões das restrições de igualdade na Hipótese
#                Nula do FBST; deve haver pelo menos uma restrição.
# inequalNHConstr: expressões das restrições de desigualdade na
#                  Hipótese Nula.
# equalDomainConstr: restrições de igualdade sob o domínio da
#                    posteriori.
# inequalDomainConstr: restrições de desigualdade sobre o domínio da
#                      posteriori.
# initialPointNH: ponto inicial sob a hipótese nula para a otimização.
# integrationExpr: expressão da posteriori a ser utilizada no
#                  cálculo da integral.
# impSampFunction: função de amostragem e densidade para o importance
#                  sampling definida pelo usuário.
#
# parâmetros de Cálculo da Evidencia contra H0:
# ---------------------------------------------
# evError, evGamma, evNDiv, evStep, evNMax, evNInit, evEchoIt
# - Ver comentários sobre a função evidence
#
# thetaMax: ponto de máximo irrestrito da posteriori.
# ...: parâmetros do otimizador ALGENCAN
#
# ==================================================================
# Retorno
# -------
# integral: Evidência contra a hipótese nula para a posteriori avaliada.
# error: medida do erro cometido no cálculo da evidência
```

```
#
# Demais valores: ver comentários sobre a função evidence
#
# ********************************************************************
# ********************************************************************
fbst <- function (optimizationExpr,
                  domainBounds,
                  equalNHConstr,
                  inequalNHConstr = NULL,
                  equalDomainConstr = NULL,
                  inequalDomainConstr = NULL,
                  initialPointNH,
                  integrationExpr = paste("exp(", optimizationExpr,")"),
                  impSampFunction,
                  indicatorMethod = FALSE,
                  evError = 0.01,
                  evGamma = 0.99,
                  evNDiv = 10,
                  evStep = 1000,
                  evNMax = 5e5,
                  evNInit = 9000,
                  evEchoIt = FALSE,
                  thetaMax = NULL,
                  ...) {
##################################################################
# Vamos procurar o ponto de máximo sob a hipótese nula (NH) #
##################################################################

# Definindo os parâmetros para a função Otimiza (adaptMLA)
# Definimos a função a ser otimizada

functn <- optimizationExpr                        # (adaptMLA)
variables <- sort(varList(optimizationExpr))      # (adaptMLA)

n <- length(variables) # numero de variaveis      # (adaptMLA)

# Testando as dimensões do dominio
if (length(domainBounds$lower) != length(domainBounds$upper))
  stop ("Os limites do dominio devem ter o mesmo comprimento")
if (length(domainBounds$lower) != n)
  stop ("O numero de variaveis do dominio nao coincide com o ",
  "numero de variaveis da funcao a ser otimizada")

bounds <- domainBounds                            # (adaptMLA)

# Definindo as restrições para encontrar o máximo da posteriori
# sob a Hipotese Nula (NH)
constraints <- c(equalNHConstr,
                 inequalNHConstr,
                 equalDomainConstr,
                 inequalDomainConstr)
m <- length(constraints) # numero total de restrições

constr <- list()
constr[['expression']] <- NULL
constr[['equality']] <- NULL
for (i in 1:m) {
  constr[['expression']][i] <- constraints[i]
  constr[['equality']][i] <-
    if (constraints[i] %in% c(equalNHConstr, equalDomainConstr))
      TRUE
```

```
    else
      FALSE
}

# Executamos a função Otimiza (adaptMLA) para buscar o ponto de máximo da
# posteriori

tst <- otimiza (f = functn,                        # (adaptMLA)
                vars = variables,                  # (adaptMLA)
                init = initialPointNH,             # (adaptMLA)
                cons = constr,                     # (adaptMLA)
                maximize = TRUE,                   # (adaptMLA)
                ...)                               # (adaptMLA)

thetaStar <- tst$par                               # (adaptMLA)

if (inherits(tst, "try-error"))                    # (adaptMLA)
  stop("Falha ao procurar ponto de maximo da posteriori sob H0")

if (is.null(thetaMax)) {

#####################################################################
# Vamos procurar o ponto de maximo irrestrito sobre o dominio #
#####################################################################

# Definindo as restrições para encontrar o maximo da posteriori
# sob o domínio
  constraints <- c(equalDomainConstr,
                   inequalDomainConstr)

  m <- length(constraints) # numero total de restrições

  constr <- list()
  if (m > 0) {
    constr[['expression']] <- NULL
    constr[['equality']] <- NULL
    for (i in 1:m) {
      constr[['expression']][i] <- constraints[i]
      constr[['equality']][i] <-
        if (constraints[i] %in% c(equalNHConstr, equalDomainConstr))
          TRUE
        else
          FALSE
    }
  }

# (obsMLA) Definição das restrições sobre o espaço paramétrico,
# a partir dos limites pré-informados

    for (i in 1:n)                                              # (adaptMLA)
     {                                                          # (adaptMLA)
       constr[['expression']][m+(2*i)-1] <- paste(variables[i],
                                          "-",bounds$lower[i]) #(adaptMLA)
       constr[['equality']][m+(2*i)-1] <- FALSE                 # (adaptMLA)
       constr[['expression']][m+(2*i)] <- paste(bounds$upper[i],
                                          "-",variables[i])     # (adaptMLA)
       constr[['equality']][m+(2*i)] <- FALSE                   # (adaptMLA)
     }

# Executamos a função Otimiza (adaptMLA) para buscar o ponto de máximo
# irrestrito da posteriori
```

```
tmx <- otimiza (f = functn,                          # (adaptMLA)
                vars = variables,                    # (adaptMLA)
                init = initialPointNH,               # (adaptMLA)
                cons = constr,                       # (adaptMLA)
                maximize = TRUE,                     # (adaptMLA)
                ...)                                 # (adaptMLA)

thetaMax <- tmx$par                                  # (adaptMLA)

if (inherits(tmx, "try-error"))                      # (adaptMLA)
  stop("Falha ao procurar ponto de maximo irrestrito da posteriori")
}

# Vamos definir dinamicamente a funcao que será usada no processo de
# integração via Monte Carlo

if (n > 1)                                           # (adaptMLA)
  {                                                  # (adaptMLA)
  cat(paste('f <- function (x) return(',
             notation(integrationExpr, mode = "R",
                      matr = TRUE),')'),
             file = 'defineF.r')
  }                                                  # (adaptMLA)
if (n == 1)                                          # (adaptMLA)
  {                                                  # (adaptMLA)
  cat(paste('f <- function (x1) {',
             integrationExpr,'}'),
             file = 'defineF.r')                     # (adaptMLA)
  }                                                  # (adaptMLA)

source ('defineF.r', local = TRUE)

ev <- evidence (f, thetaStar, thetaMax = thetaMax, impSampFunction,
                indicatorMethod = indicatorMethod,
                error = evError, gamma = evGamma, nDiv = evNDiv,
                step = evStep, nMax = evNMax, nInit = evNInit,
                echoIt = evEchoIt)

return (ev)
}

##########################################################################
```

A.3. Biblioteca sfbst.R

Observação: Essa e as demais bibliotecas se encontram disponíveis para *download* na página http://www.chancedegol.com.br/sfbst/sfbst.htm .

```
# ====================================================================
# Arquivo: sfbst.r
# ====================================================================
# Autor: Marcelo Leme de Arruda
# ====================================================================
#
# ====================================================================
# Módulo: Rotinas para execução do teste FBST Seqüencial
# para distribuições conjugadas.
#
# Last update of any of the component of this module:
#
# January 31, 2012.
#
########################################################################

source("fbst.r")

# ====================================================================
#
# Função: sfbst.betabin
#
# ====================================================================
# Descrição
# ---------
# Função que executa o FBST Seqüencial para verossimilhanças
# Bernoulli, Binomial, Geométrica ou Binomial Negativa, com
# prioris/posterioris Beta.
#
# ====================================================================
# Uso
# ---
# sfbst.betabin (alpha,
#                beta,
#                parambounds,
#                equalNHConstr,
#                inequalNHConstr = NULL,
#                equalDomainConstr = NULL,
#                inequalDomainConstr = NULL,
#                initialPoint,
#                impSampFunction,
#                alphacut,
#                rstar,
#                abc = NULL,
#                custo,
#                epsilon = 0.0001,
#
#                indicatorMethod = FALSE,
#                error = 0.01,
#                gamma = 0.99,
#                nDiv = 10,
#                step = 100,
#                nMax = 5e5,
#                nInit = 10000,
#                echoIt = FALSE)
```

```
# ================================================================
# Argumentos
# ----------
# alpha: primeiro parâmetro (alfa) da posteriori Beta(alfa,beta)
# beta: segundo parâmetro (beta) da posteriori Beta(alfa,beta)
# paramBounds: lista de limitantes inferior e superior do espaço
#              paramétrico. Em caso de limitantes infinitos, definir
#              um valor suficientemente grande, mas limitado, para
#              evitar problemas com o otimizador.
# equalNHConstr: expressões das restrições de igualdade na Hipótese
#                Nula; deve haver pelo menos uma restrição.
# inequalNHConstr: expressões das restrições de desigualdade na Hipótese
#                  Nula.
# equalDomainConstr: restrições de igualdade sob o dominio da posteriori.
# inequalDomainConstr: restrições de desigualdade sobre o dominio da
#                      posteriori.
# initialPoint: ponto inicial para a otimização.
# impSampFunction: função de amostragem e densidade para o importance
#                  sampling, definida pelo usuário.
# alphacut: valor de corte (i.e. valor alpha que para Ev > alpha,
#           aceita-se H0 e para Ev < alpha, rejeita-se H0.
# rStar: risco (perda) máximo que se pode sofrer ao tomar a decisão ótima
# abc: vetor de parâmetros das funções de perda, caso o usuário queira
#      informá-los diretamente.
# custo: custo (unitário) de cada observação realizada.
# epsilon: critério de parada para as amostragens simuladas no processo
#
# parâmetros para execução do fbst: ver descrição e comentários
#                                   no arquivo fbst.r
# ================================================================
# Retorno
# -------
# integral: Evidência A FAVOR da hipótese nula para a posteriori avaliada.
# error: medida do erro cometido no cálculo da Evidência
# sample.size: quantidade de itens a serem amostrados
# riscos: Riscos Totais (incluído o custo de amostragem) de tomada da
#         decisão imediatamente e após k = 1, 2, ..., sample.size+1
#         observações serem amostradas
#
# Demais valores: ver comentários sobre a função evidence
#
# ****************************************************************
# ****************************************************************

sfbst.betabin <- function (alpha,
                           beta,
                           paramBounds,
                           equalNHConstr,
                           inequalNHConstr = NULL,
                           equalDomainConstr = NULL,
                           inequalDomainConstr = NULL,
                           initialPoint,
                           impSampFunction,
                           alphacut,
                           rstar,
                           abc = NULL,
                           custo,
                           epsilon = 0.0001,
                           ...) {

if (is.null(abc)) { a <- rstar/alphacut } else { a <- abc[1] }
```

```
if (is.null(abc)) { b <- 0 } else { b <- abc[2] }
if (is.null(abc)) { c <- rstar/(1-alphacut) } else { c <- abc[3] }

robarra <- vector()
robarra[1] <- NULL

# Construindo a densidade (na verdade o seu log) da posteriori a
# ser avaliada

fposteriori <- paste("(",alpha-1,"*log(x1)) + (",beta-1,"*log(1-x1))")

# Executando a "zerésima" iteração do processo

evbarra <- fbst(fposteriori, paramBounds, equalNHConstr, inequalNHConstr,
                equalDomainConstr, inequalDomainConstr, initialPoint,
                impSampFunction = impSampFunction, ...)

ev0 <- 1 - evbarra$integral
robarra0 <- min(a*ev0 , b+(c*(1-ev0)))
m0 <- robarra0
riscos <- vector()
riscos[1] <- robarra0
names(riscos)[1] <- 0

k <- 0
while (1)
  {
  k <- k+1
  ro <- vector()
  ro[1] <- NULL
  i <- 0
  robarra[i] <- 0
  while(1)
    {
    i <- i+1
    theta.i <- rbeta(1, alpha, beta)      # gera um valor theta.i com
                                          # distribuição Beta(alpha,beta)
    xsample <- rbinom(k,1,theta.i)        # gera k observações com
                                          # distribuição Bernoulli(theta.i)

# atualizando o log da densidade posteriori em função das k observações
# geradas
    alpha.i <- alpha + sum(xsample)
    beta.i <- beta + k - sum(xsample)
    posteriori.i <- paste("(",alpha.i-1,"*log(x1)) + (",beta.i-1,
                          "*log(1-x1))")

    evbarra.i <- fbst(posteriori.i, paramBounds, equalNHConstr,
                      inequalNHConstr, equalDomainConstr,
                      inequalDomainConstr, initialPoint,
                      impSampFunction = impSampFunction, ...)
    ev.i <- 1 - evbarra.i$integral
    ro[i] <- min(a*ev.i , b+(c*(1-ev.i)))
    robarra[i] <- mean(ro)

    difro <- 0
    if (i > 1) difro <- abs(robarra[i] - robarra[i-1])/robarra[i-1]
    if ((i > 1) && (difro <= epsilon))
      {
      robarra.k <- robarra[i]
      riscos[k+1] <- robarra.k + (k*custo)
```

```
      names(riscos)[k+1] <- k
      break
      }
    }
  if ((robarra.k + (k*custo)) >= m0) break
  m0 <- min(m0 , robarra.k + (k*custo))
  }
evbarra[['sample.size']] <- k-1
evbarra$riscos <- riscos
evbarra$integral <- 1 - evbarra$integral
return(evbarra)
}

#############################################################################
# ===========================================================
#
# Função: sfbst.gamapois
#
# ===========================================================
# Descrição
# ---------
# Função que executa o FBST Sequencial para verossimilhanças
# Poisson, com prioris/posterioris Gama.
#
# ===========================================================
# Uso
# ---
# sfbst.gamapois (alpha,
#                 beta,
#                 parambounds,
#                 equalNHConstr,
#                 inequalNHConstr = NULL,
#                 equalDomainConstr = NULL,
#                 inequalDomainConstr = NULL,
#                 initialPoint,
#                 impSampFunction,
#                 alphacut,
#                 rstar,
#                 abc = NULL,
#                 custo,
#                 epsilon = 0.0001,
#
#                 indicatorMethod = FALSE,
#                 error = 0.01,
#                 gamma = 0.99,
#                 nDiv = 10,
#                 step = 100,
#                 nMax = 5e5,
#                 nInit = 10000,
#                 echoIt = FALSE)
# ===========================================================
# Argumentos
# ----------
# alpha: parâmetro de forma (alfa) da posteriori Gama(alfa,beta)
# beta: parâmetro de escala (beta) da posteriori Gama(alfa,beta)
# paramBounds: lista de limitantes inferior e superior do espaço
#              paramétrico. Em caso de limitantes infinitos, definir
#              um valor suficientemente grande, mas limitado, para
#              evitar problemas com o otimizador.
# equalNHConstr: expressões das restrições de igualdade na Hipótese
#                Nula; deve haver pelo menos uma restrição.
```

```
# inequalNHConstr: expressões das restrições de desigualdade na Hipótese
#                  Nula.
# equalDomainConstr: restrições de igualdade sob o dominio da posteriori.
# inequalDomainConstr: restrições de desigualdade sobre o dominio da
#                      posteriori.
# initialPoint: ponto inicial para a otimização.
# impSampFunction: função de amostragem e densidade para o importance
#                  sampling, definida pelo usuário.
# alphacut: valor de corte (i.e. valor alpha que para Ev > alpha,
#           aceita-se H0 e para Ev < alpha, rejeita-se H0.
# rStar: risco (perda) máximo que se pode sofrer ao tomar a decisão ótima
# abc: vetor de parâmetros das funções de perda, caso o usuário queira
#      informá-los diretamente.
# custo: custo (unitário) de cada observação realizada.
# epsilon: critério de parada para as amostragens simuladas no processo
#
# parâmetros para execução do fbst: ver descrição e comentários
#                                   no arquivo fbst.r
# ================================================================
# Retorno
# -------
# integral: Evidência A FAVOR da hipótese nula para a posteriori avaliada.
# error: medida do erro cometido no cálculo da Evidência
# sample.size: quantidade de itens a serem amostrados
# riscos: Riscos Totais (incluído o custo de amostragem) de tomada da
#         decisão imediatamente e após k = 1, 2, ..., sample.size+1
#         observações serem amostradas
#
# Demais valores: ver comentários sobre a função evidence
#
# ****************************************************************
# ****************************************************************

sfbst.gamapois <- function (alpha,
                            beta,
                            paramBounds,
                            equalNHConstr,
                            inequalNHConstr = NULL,
                            equalDomainConstr = NULL,
                            inequalDomainConstr = NULL,
                            initialPoint,
                            impSampFunction,
                            alphacut,
                            rstar,
                            abc = NULL,
                            custo,
                            epsilon = 0.0001,
                            ...) {

if (is.null(abc)) { a <- rstar/alphacut } else { a <- abc[1] }
if (is.null(abc)) { b <- 0 } else { b <- abc[2] }
if (is.null(abc)) { c <- rstar/(1-alphacut) } else { c <- abc[3] }

robarra <- vector()
robarra[1] <- NULL

# Construindo a densidade (na verdade o seu log) da posteriori a ser
# avaliada

fposteriori <- paste("(",alpha - 1,"* log(x1)) - (",beta," * x1)")
```

```
# Executando a "zerésima" iteração do processo

evbarra <- fbst(fposteriori, paramBounds, equalNHConstr, inequalNHConstr,
                equalDomainConstr, inequalDomainConstr, initialPoint,
                impSampFunction = impSampFunction, ...)

ev0 <- 1 - evbarra$integral
robarra0 <- min(a*ev0 , b+(c*(1-ev0)))
m0 <- robarra0
riscos <- vector()
riscos[1] <- robarra0
names(riscos)[1] <- 0

k <- 0
while (1)
  {
  k <- k+1
  ro <- vector()
  ro[1] <- NULL
  i <- 0
  robarra[i] <- 0
  while(1)
    {
    i <- i+1
    theta.i <- rgamma(1, alpha, scale = beta) # gera um valor theta.i com
                                              # distrib. Gama(alpha,beta)
    xsample <- rpois(k,theta.i)               # gera k observações com
                                              # distrib. Poisson(theta.i)

# atualizando o log da densidade posteriori em função das k observações
# geradas
    alpha.i <- alpha + sum(xsample)
    beta.i <- beta + k
    posteriori.i <- paste("(",alpha.i - 1,"* log(x1)) - (",beta.i," * x1)")

    evbarra.i <- fbst(posteriori.i, paramBounds, equalNHConstr,
                      inequalNHConstr, equalDomainConstr,
                      inequalDomainConstr, initialPoint,
                      impSampFunction = impSampFunction, ...)

    ev.i <- 1 - evbarra.i$integral
    ro[i] <- min(a*ev.i , b+(c*(1-ev.i)))
    robarra[i] <- mean(ro)

    difro <- 0
    if (i > 1) difro <- abs(robarra[i] - robarra[i-1])/robarra[i-1]
    if ((i > 1) && (difro <= epsilon))
      {
      robarra.k <- robarra[i]
      riscos[k+1] <- robarra.k + (k*custo)
      names(riscos)[k+1] <- k
      break
      }
    }
  if ((robarra.k + (k*custo)) >= m0) break
  m0 <- min(m0 , robarra.k + (k*custo))
  }
evbarra[['sample.size']] <- k-1
evbarra$riscos <- riscos
evbarra$integral <- 1 - evbarra$integral
return(evbarra)
```

```
}

#############################################################################
# ===========================================================
#
# Função: sfbst.gamaexp
#
# ===========================================================
# Descrição
# ---------
# Função que executa o FBST Sequencial para verossimilhanças
# Exponencial, com prioris/posterioris Gama.
#
# ===========================================================
# Uso
# ---
# sfbst.gamaexp (alpha,
#                beta,
#                parambounds,
#                equalNHConstr,
#                inequalNHConstr = NULL,
#                equalDomainConstr = NULL,
#                inequalDomainConstr = NULL,
#                initialPoint,
#                impSampFunction,
#                alphacut,
#                rstar,
#                abc = NULL,
#                custo,
#                epsilon = 0.0001,
#
#                indicatorMethod = FALSE,
#                error = 0.01,
#                gamma = 0.99,
#                nDiv = 10,
#                step = 100,
#                nMax = 5e5,
#                nInit = 10000,
#                echoIt = FALSE)
# ===========================================================
# Argumentos
# ----------
# alpha: parâmetro de forma (alfa) da posteriori Gama(alfa,beta)
# beta: parâmetro de escala (beta) da posteriori Gama(alfa,beta)
# paramBounds: lista de limitantes inferior e superior do espaço
#              paramétrico. Em caso de limitantes infinitos, definir
#              um valor suficientemente grande, mas limitado, para
#              evitar problemas com o otimizador.
# equalNHConstr: expressões das restrições de igualdade na Hipótese
#                Nula; deve haver pelo menos uma restrição.
# inequalNHConstr: expressões das restrições de desigualdade na Hipótese
#                   Nula.
# equalDomainConstr: restrições de igualdade sob o dominio da posteriori.
# inequalDomainConstr: restrições de desigualdade sobre o dominio da
#                      posteriori.
# initialPoint: ponto inicial para a otimização.
# impSampFunction: função de amostragem e densidade para o importance
#                  sampling, definida pelo usuário.
# alphacut: valor de corte (i.e. valor alpha que para Ev > alpha,
#           aceita-se H0 e para Ev < alpha, rejeita-se H0.
# rStar: risco (perda) máximo que se pode sofrer ao tomar a decisão ótima
```

```
# abc: vetor de parâmetros das funções de perda, caso o usuário queira
#      informá-los diretamente.
# custo: custo (unitário) de cada observação realizada.
# epsilon: critério de parada para as amostragens simuladas no processo
#
# parâmetros para execução do fbst: ver descrição e comentários
#                                   no arquivo fbst.r
# ================================================================
# Retorno
# -------
# integral: Evidência A FAVOR da hipótese nula para a posteriori avaliada.
# error: medida do erro cometido no cálculo da Evidência
# sample.size: quantidade de itens a serem amostrados
# riscos: Riscos Totais (incluído o custo de amostragem) de tomada da
#         decisão imediatamente e após k = 1, 2, ..., sample.size+1
#         observações serem amostradas
#
# Demais valores: ver comentários sobre a função evidence
#
# ****************************************************************
# ****************************************************************

sfbst.gamaexp <- function (alpha,
                           beta,
                           paramBounds,
                           equalNHConstr,
                           inequalNHConstr = NULL,
                           equalDomainConstr = NULL,
                           inequalDomainConstr = NULL,
                           initialPoint,
                           impSampFunction,
                           alphacut,
                           rstar,
                           abc = NULL,
                           custo,
                           epsilon = 0.0001,
                           ...) {

if (is.null(abc)) { a <- rstar/alphacut } else { a <- abc[1] }
if (is.null(abc)) { b <- 0 } else { b <- abc[2] }
if (is.null(abc)) { c <- rstar/(1-alphacut) } else { c <- abc[3] }

robarra <- vector()
robarra[1] <- NULL

# Construindo a densidade (na verdade o seu log) da posteriori a ser
# avaliada

fposteriori <- paste("(",alpha - 1,"* log(x1)) - (",beta," * x1)")

# Executando a "zerésima" iteração do processo

evbarra <- fbst(fposteriori, paramBounds, equalNHConstr, inequalNHConstr,
                equalDomainConstr, inequalDomainConstr, initialPoint,
                impSampFunction = impSampFunction, ...)

ev0 <- 1 - evbarra$integral
robarra0 <- min(a*ev0 , b+(c*(1-ev0)))
m0 <- robarra0
riscos <- vector()
riscos[1] <- robarra0
```

```
names(riscos)[1] <- 0

k <- 0
while (1)
  {
  k <- k+1
  ro <- vector()
  ro[1] <- NULL
  i <- 0
  robarra[i] <- 0
  while(1)
    {
    i <- i+1
    theta.i <- rgamma(1, alpha, scale = beta) # gera um valor theta.i com
                                              # distrib. Gama(alpha,beta)
    xsample <- rexp(k,theta.i)                # gera k observações com
                                              # distr. Exponencial(theta.i)

# atualizando o log da densidade posteriori em função das k observações
# geradas
    alpha.i <- alpha + k
    beta.i <- beta + sum(xsample)
    posteriori.i <- paste("(",alpha.i - 1,"* log(x1)) - (",beta.i," * x1)")

    evbarra.i <- fbst(posteriori.i, paramBounds, equalNHConstr,
                      inequalNHConstr, equalDomainConstr,
                      inequalDomainConstr, initialPoint,
                      impSampFunction = impSampFunction, ...)
    ev.i <- 1 - evbarra.i$integral
    ro[i] <- min(a*ev.i , b+(c*(1-ev.i)))
    robarra[i] <- mean(ro)

    difro <- 0
    if (i > 1) difro <- abs(robarra[i] - robarra[i-1])/robarra[i-1]
    if ((i > 1) && (difro <= epsilon))
      {
      robarra.k <- robarra[i]
      riscos[k+1] <- robarra.k + (k*custo)
      names(riscos)[k+1] <- k
      break
      }
    }
  if ((robarra.k + (k*custo)) >= m0) break
  m0 <- min(m0 , robarra.k + (k*custo))
  }
evbarra[['sample.size']] <- k-1
evbarra$riscos <- riscos
evbarra$integral <- 1 - evbarra$integral
return(evbarra)
}


#############################################################################
# ===========================================================
#
# Função: sfbst.normnorm
#
# ===========================================================
# Descrição
# ---------
# Função que executa o FBST Sequencial para verossimilhanças
```

```
# Normal, com variância conhecida, com prioris/posterioris (para
# a média) Normal.
#
# ================================================================
# Uso
# ---
# sfbst.normnorm (mu,
#                 varp,
#                 varv,
#                 parambounds,
#                 equalNHConstr,
#                 inequalNHConstr = NULL,
#                 equalDomainConstr = NULL,
#                 inequalDomainConstr = NULL,
#                 initialPoint,
#                 impSampFunction,
#                 alphacut,
#                 rstar,
#                 abc = NULL,
#                 custo,
#                 epsilon = 0.0001,
#
#                 indicatorMethod = FALSE,
#                 error = 0.01,
#                 gamma = 0.99,
#                 nDiv = 10,
#                 step = 100,
#                 nMax = 5e5,
#                 nInit = 10000,
#                 echoIt = FALSE)
# ================================================================
# Argumentos
# ----------
# mu: média da posteriori Normal (mu,varp)
# varp variância da posteriori Normal (mu,varp)
# varv: variância da verossimilhança Normal (theta,varv)
# paramBounds: lista de limitantes inferior e superior do espaço
#              paramétrico. Em caso de limitantes infinitos, definir
#              um valor suficientemente grande, mas limitado, para
#              evitar problemas com o otimizador.
# equalNHConstr: expressões das restrições de igualdade na Hipótese
#                Nula; deve haver pelo menos uma restrição.
# inequalNHConstr: expressões das restrições de desigualdade na Hipótese
#                  Nula.
# equalDomainConstr: restrições de igualdade sob o dominio da posteriori.
# inequalDomainConstr: restrições de desigualdade sobre o dominio da
#                      posteriori.
# initialPoint: ponto inicial para a otimização.
# impSampFunction: função de amostragem e densidade para o importance
#                  sampling, definida pelo usuário.
# alphacut: valor de corte (i.e. valor alpha que para Ev > alpha,
#           aceita-se H0 e para Ev < alpha, rejeita-se H0.
# rStar: risco (perda) máximo que se pode sofrer ao tomar a decisão ótima
# abc: vetor de parâmetros das funções de perda, caso o usuário queira
#      informá-los diretamente.
# custo: custo (unitário) de cada observação realizada.
# epsilon: critério de parada para as amostragens simuladas no processo
#
# parâmetros para execução do fbst: ver descrição e comentários
#                                   no arquivo fbst.r
# ================================================================
```

```
# Retorno
# -------
# integral: Evidência A FAVOR da hipótese nula para a posteriori avaliada.
# error: medida do erro cometido no cálculo da Evidência
# sample.size: quantidade de itens a serem amostrados
# riscos: Riscos Totais (incluído o custo de amostragem) de tomada da
#         decisão imediatamente e após k = 1, 2, ..., sample.size+1
#         observações serem amostradas
#
# Demais valores: ver comentários sobre a função evidence
#
# ******************************************************************
# ******************************************************************

sfbst.normnorm <- function (mu,
                            varp,
                            varv,
                            paramBounds,
                            equalNHConstr,
                            inequalNHConstr = NULL,
                            equalDomainConstr = NULL,
                            inequalDomainConstr = NULL,
                            initialPoint,
                            impSampFunction,
                            alphacut,
                            rstar,
                            abc = NULL,
                            custo,
                            epsilon = 0.0001,
                            ...) {

if (is.null(abc)) { a <- rstar/alphacut } else { a <- abc[1] }
if (is.null(abc)) { b <- 0 } else { b <- abc[2] }
if (is.null(abc)) { c <- rstar/(1-alphacut) } else { c <- abc[3] }

robarra <- vector()
robarra[1] <- NULL

# Construindo a densidade (na verdade o seu log) da posteriori a ser
# avaliada

fposteriori <- paste("-((x1 - ",mu,")^2) / ",2 * varp)

# Executando a "zerésima" iteração do processo

evbarra <- fbst(fposteriori, paramBounds, equalNHConstr, inequalNHConstr,
                equalDomainConstr, inequalDomainConstr, initialPoint,
                impSampFunction = impSampFunction, ...)

ev0 <- 1 - evbarra$integral
robarra0 <- min(a*ev0 , b+(c*(1-ev0)))
m0 <- robarra0
riscos <- vector()
riscos[1] <- robarra0
names(riscos)[1] <- 0

k <- 0
while (1)
  {
  k <- k+1
  ro <- vector()
```

```
  ro[1] <- NULL
  i <- 0
  robarra[i] <- 0
  while(1)
    {
    i <- i+1
    theta.i <- rnorm(1, mu, sqrt(varp))       # gera um valor theta.i com
                                              # distrib. Normal(mu, varp)
    xsample <- rnorm(k, theta.i, sqrt(varv))  # gera k observações com
                                              # dist. Normal(theta.i, varv)

# atualizando o log da densidade posteriori em função das k observações
# geradas
    mu.i <- ((mu/varp) + (sum(xsample)/varv)) / ((1/varp) + (k/varv))
    varp.i <- 1 / ((1/varp) + (k/varv))
    posteriori.i <- paste("-((x1 - ",mu.i,")^2) / ",2 * varp.i)

    evbarra.i <- fbst(posteriori.i, paramBounds, equalNHConstr,
                      inequalNHConstr, equalDomainConstr,
                      inequalDomainConstr, initialPoint,
                      impSampFunction = impSampFunction, ...)
    ev.i <- 1 - evbarra.i$integral
    ro[i] <- min(a*ev.i , b+(c*(1-ev.i)))
    robarra[i] <- mean(ro)
    difro <- 0
    if (i > 1) difro <- abs(robarra[i] - robarra[i-1])/robarra[i-1]
    if ((i > 1) && (difro <= epsilon))
      {
      robarra.k <- robarra[i]
      riscos[k+1] <- robarra.k + (k*custo)
      names(riscos)[k+1] <- k
      break
      }
    }
  if ((robarra.k + (k*custo)) >= m0) break
  m0 <- min(m0 , robarra.k + (k*custo))
  }
evbarra[['sample.size']] <- k-1
evbarra$riscos <- riscos
evbarra$integral <- 1 - evbarra$integral
return(evbarra)
}

#############################################################################
# ===========================================================
#
# Função: sfbst.gamanorm
#
# ===========================================================
# Descrição
# ---------
# Função que executa o FBST Sequencial para verossimilhanças
# Normal, com média conhecida, com prioris/posterioris (para
# a precisão = 1/variância) Gama.
#
# ===========================================================
# Uso
# ---
# sfbst.gamanorm (alpha,
#                 beta,
#                 mu,
```

```
#                 parambounds,
#                 equalNHConstr,
#                 inequalNHConstr = NULL,
#                 equalDomainConstr = NULL,
#                 inequalDomainConstr = NULL,
#                 initialPoint,
#                 impSampFunction,
#                 alphacut,
#                 rstar,
#                 abc = NULL,
#                 custo,
#                 epsilon = 0.0001,
#
#                 indicatorMethod = FALSE,
#                 error = 0.01,
#                 gamma = 0.99,
#                 nDiv = 10,
#                 step = 100,
#                 nMax = 5e5,
#                 nInit = 10000,
#                 echoIt = FALSE)
# ================================================================
# Argumentos
# ----------
# alpha: parâmetro de forma (alfa) da posteriori Gama(alfa,beta)
# beta: parâmetro de escala (beta) da posteriori Gama(alfa,beta)
# mu: média da verossimilhança Normal (mu,theta)
# paramBounds: lista de limitantes inferior e superior do espaço
#              paramétrico. Em caso de limitantes infinitos, definir
#              um valor suficientemente grande, mas limitado, para
#              evitar problemas com o otimizador.
# equalNHConstr: expressões das restrições de igualdade na Hipótese
#                Nula; deve haver pelo menos uma restrição.
# inequalNHConstr: expressões das restrições de desigualdade na Hipótese
#                  Nula.
# equalDomainConstr: restrições de igualdade sob o dominio da posteriori.
# inequalDomainConstr: restrições de desigualdade sobre o dominio da
#                      posteriori.
# initialPoint: ponto inicial para a otimização.
# impSampFunction: função de amostragem e densidade para o importance
#                  sampling, definida pelo usuário.
# alphacut: valor de corte (i.e. valor alpha que para Ev > alpha,
#           aceita-se H0 e para Ev < alpha, rejeita-se H0.
# rStar: risco (perda) máximo que se pode sofrer ao tomar a decisão ótima
# abc: vetor de parâmetros das funções de perda, caso o usuário queira
#      informá-los diretamente.
# custo: custo (unitário) de cada observação realizada.
# epsilon: critério de parada para as amostragens simuladas no processo
#
# parâmetros para execução do fbst: ver descrição e comentários
#                                   no arquivo fbst.r
# ================================================================
# Retorno
# -------
# integral: Evidência A FAVOR da hipótese nula para a posteriori avaliada.
# error: medida do erro cometido no cálculo da Evidência
# sample.size: quantidade de itens a serem amostrados
# riscos: Riscos Totais (incluído o custo de amostragem) de tomada da
#         decisão imediatamente e após k = 1, 2, ..., sample.size+1
#         observações serem amostradas
#
```

```
# Demais valores: ver comentários sobre a função evidence
#
# ********************************************************************
# ********************************************************************

sfbst.gamanorm <- function (alpha,
                            beta,
                            mu,
                            paramBounds,
                            equalNHConstr,
                            inequalNHConstr = NULL,
                            equalDomainConstr = NULL,
                            inequalDomainConstr = NULL,
                            initialPoint,
                            impSampFunction,
                            alphacut,
                            rstar,
                            abc = NULL,
                            custo,
                            epsilon = 0.0001,
                            ...) {

if (is.null(abc)) { a <- rstar/alphacut } else { a <- abc[1] }
if (is.null(abc)) { b <- 0 } else { b <- abc[2] }
if (is.null(abc)) { c <- rstar/(1-alphacut) } else { c <- abc[3] }

robarra <- vector()
robarra[1] <- NULL

# Construindo a densidade (na verdade o seu log) da posteriori a ser
avaliada

fposteriori <- paste("(",alpha - 1,"* log(x1)) - (",beta," * x1)")

# Executando a "zerésima" iteração do processo

evbarra <- fbst(fposteriori, paramBounds, equalNHConstr, inequalNHConstr,
                equalDomainConstr, inequalDomainConstr, initialPoint,
                impSampFunction = impSampFunction, ...)

ev0 <- 1 - evbarra$integral
robarra0 <- min(a*ev0 , b+(c*(1-ev0)))
m0 <- robarra0
riscos <- vector()
riscos[1] <- robarra0
names(riscos)[1] <- 0

k <- 0
while (1)
  {
  k <- k+1
  ro <- vector()
  ro[1] <- NULL
  i <- 0
  robarra[i] <- 0
  while(1)
    {
    i <- i+1
    theta.i <- rgamma(1, alpha, scale = beta) # gera um valor theta.i com
                                              # distrib. Gama(alpha,beta)
    xsample <- rnorm(k, mu, sqrt(1/theta.i))  # gera k observações com
```

```
                                                  # dist. Normal(mu, 1/theta.i)
# atualizando o log da densidade posteriori em função das k observações
# geradas
    alpha.i <- alpha + (k/2)
    if (k == 1) beta.i <- beta
    if (k > 1) beta.i <- beta + ((k-1)*var(xsample)/2)
    posteriori.i <- paste("(",alpha.i - 1,"* log(x1)) - (",beta.i," * x1)")

    evbarra.i <- fbst(posteriori.i, paramBounds, equalNHConstr,
                      inequalNHConstr, equalDomainConstr,
                      inequalDomainConstr, initialPoint,
                      impSampFunction = impSampFunction, ...)
    ev.i <- 1 - evbarra.i$integral
    ro[i] <- min(a*ev.i , b+(c*(1-ev.i)))
    robarra[i] <- mean(ro)
    difro <- 0
    if (i > 1) difro <- abs(robarra[i] - robarra[i-1])/robarra[i-1]
    if ((i > 1) && (difro <= epsilon))
      {
      robarra.k <- robarra[i]
      riscos[k+1] <- robarra.k + (k*custo)
      names(riscos)[k+1] <- k
      break
      }
    }
  if ((robarra.k + (k*custo)) >= m0) break
  m0 <- min(m0 , robarra.k + (k*custo))
  }
evbarra[['sample.size']] <- k-1
evbarra$riscos <- riscos
evbarra$integral <- 1 - evbarra$integral
return(evbarra)
}

#############################################################################
# ===========================================================
#
# Função: sfbst.dirimult
#
# ===========================================================
# Descrição
# ---------
# Função que executa o FBST Sequencial para verossimilhanças
# Multinomial, com prioris/posterioris Dirichlet (com dimensão
# menor que 10).
#
# ===========================================================
# Uso
# ---
# sfbst.dirimult (alpha,
#                 parambounds,
#                 equalNHConstr,
#                 inequalNHConstr = NULL,
#                 equalDomainConstr = NULL,
#                 inequalDomainConstr = NULL,
#                 initialPoint,
#                 impSampFunction,
#                 alphacut,
#                 rstar,
#                 abc = NULL,
#                 custo,
```

```
#                  epsilon = 0.0001,
#
#                  indicatorMethod = FALSE,
#                  error = 0.01,
#                  gamma = 0.99,
#                  nDiv = 10,
#                  step = 100,
#                  nMax = 5e5,
#                  nInit = 10000,
#                  echoIt = FALSE)
# ================================================================
# Argumentos
# ----------
# alpha: vetor de parâmetros da posteriori Dirichlet(alpha)
# paramBounds: lista de limitantes inferior e superior do espaço
#              paramétrico. Em caso de limitantes infinitos, definir
#              um valor suficientemente grande, mas limitado, para
#              evitar problemas com o otimizador.
# equalNHConstr: expressões das restrições de igualdade na Hipótese
#                Nula; deve haver pelo menos uma restrição.
# inequalNHConstr: expressões das restrições de desigualdade na Hipótese
#                  Nula.
# equalDomainConstr: restrições de igualdade sob o dominio da posteriori.
# inequalDomainConstr: restrições de desigualdade sobre o dominio da
#                      posteriori.
# initialPoint: ponto inicial para a otimização.
# impSampFunction: função de amostragem e densidade para o importance
#                  sampling, definida pelo usuário.
# alphacut: valor de corte (i.e. valor alpha que para Ev > alpha,
#           aceita-se H0 e para Ev < alpha, rejeita-se H0.
# rStar: risco (perda) máximo que se pode sofrer ao tomar a decisão ótima
# abc: vetor de parâmetros das funções de perda, caso o usuário queira
#      informá-los diretamente.
# custo: custo (unitário) de cada observação realizada.
# epsilon: critério de parada para as amostragens simuladas no processo
#
# parâmetros para execução do fbst: ver descrição e comentários
#                                   no arquivo fbst.r
# ================================================================
# Retorno
# -------
# integral: Evidência A FAVOR da hipótese nula para a posteriori avaliada.
# error: medida do erro cometido no cálculo da Evidência
# sample.size: quantidade de itens a serem amostrados
# riscos: Riscos Totais (incluído o custo de amostragem) de tomada da
#         decisão imediatamente e após k = 1, 2, ..., sample.size+1
#         observações serem amostradas
#
# Demais valores: ver comentários sobre a função evidence
#
# ****************************************************************
# ****************************************************************

sfbst.dirimult <- function (alpha,
                            paramBounds,
                            equalNHConstr,
                            inequalNHConstr = NULL,
                            equalDomainConstr = NULL,
                            inequalDomainConstr = NULL,
                            initialPoint,
                            impSampFunction,
```

```
                              alphacut,
                              rstar,
                              abc = NULL,
                              custo,
                              epsilon = 0.0001,
                              ...) {

if (is.null(abc)) { a <- rstar/alphacut } else { a <- abc[1] }
if (is.null(abc)) { b <- 0 } else { b <- abc[2] }
if (is.null(abc)) { c <- rstar/(1-alphacut) } else { c <- abc[3] }

varx <- c("x1","x2","x3","x4","x5","x6","x7","x8","x9")

robarra <- vector()
robarra[1] <- NULL

# Construindo a densidade (na verdade o seu log) da posteriori a ser
# avaliada

fposteriori <- paste("(",alpha[1] - 1,"* log(x1))")
for (i in 2:length(alpha))
  {
  fposteriori <- paste(fposteriori,"+ (",alpha[i] - 1,"*
log(",varx[i],"))")
  }


# Executando a "zerésima" iteração do processo

evbarra <- fbst(fposteriori, paramBounds, equalNHConstr, inequalNHConstr,
                equalDomainConstr, inequalDomainConstr, initialPoint,
                impSampFunction = impSampFunction, ...)

ev0 <- 1 - evbarra$integral
robarra0 <- min(a*ev0 , b+(c*(1-ev0)))
m0 <- robarra0
riscos <- vector()
riscos[1] <- robarra0
names(riscos)[1] <- 0

k <- 0
while (1)
  {
  k <- k+1
  theta.i <- vector()
  ro <- vector()
  ro[1] <- NULL
  i <- 0
  robarra[i] <- 0
  while(1)
    {
    i <- i+1
    for (j in 1:length(alpha))
      {
      theta.i[j] <- rgamma(1,alpha[j],1)    # gera um vetor theta.i com
      }                                     # distribuição Dirichlet(alpha)
    theta.i <- theta.i/sum(theta.i)
    xsample <- rep(0,length(alpha))
    thetacum <- cumsum(theta.i)
    for (j in 1:k)
       {
```

```
       x <- runif(1,0,1)
       y <- as.numeric(thetacum < x)               # gera k observações
       xsample[sum(y)+1] <- xsample[sum(y)+1] + 1  # com distribuição
       }                                           # Multinomial(theta.i)

# atualizando o log da densidade posteriori em função das k observações
# geradas
    alpha.i <- alpha + xsample
    posteriori.i <- paste("(",alpha.i[1] - 1,"* log(x1))")
    for (j in 2:length(alpha.i))
      {
      posteriori.i <- paste(posteriori.i,"+ (",alpha.i[j] - 1,
                            "* log(",varx[j],"))")
      }

    evbarra.i <- fbst(posteriori.i, paramBounds, equalNHConstr,
                      inequalNHConstr, equalDomainConstr,
                      inequalDomainConstr, initialPoint,
                      impSampFunction = impSampFunction, ...)
    ev.i <- 1 - evbarra.i$integral
    ro[i] <- min(a*ev.i , b+(c*(1-ev.i)))
    robarra[i] <- mean(ro)
    difro <- 0
    if (i > 1) difro <- abs(robarra[i] - robarra[i-1])/robarra[i-1]
    if ((i > 1) && (difro <= epsilon))
      {
      robarra.k <- robarra[i]
      riscos[k+1] <- robarra.k + (k*custo)
      names(riscos)[k+1] <- k
      break
      }
    }
  if ((robarra.k + (k*custo)) >= m0) break
  m0 <- min(m0 , robarra.k + (k*custo))
  }
evbarra[['sample.size']] <- k-1
evbarra$riscos <- riscos
evbarra$integral <- 1 - evbarra$integral
return(evbarra)
}


#############################################################################
# ===================================================================
#
# Função: sfbst.normgama
#
# ===================================================================
# Descrição
# ---------
# Função que executa o FBST Sequencial para verossimilhanças
# Normal (com média E variância desconhecidas) com prioris/posterioris
# Normal-Gama.
#
# ===================================================================
# Uso
# ---
# sfbst.normgama (mu,
#                 tau,
#                 alpha,
#                 beta,
```

```
#                 equalNHConstr,
#                 inequalNHConstr = NULL,
#                 equalDomainConstr = NULL,
#                 inequalDomainConstr = NULL,
#                 initialPoint,
#                 impSampFunction,
#                 alphacut,
#                 rstar,
#                 abc = NULL,
#                 custo,
#                 epsilon = 0.0001,
#
#                 indicatorMethod = FALSE,
#                 error = 0.01,
#                 gamma = 0.99,
#                 nDiv = 10,
#                 step = 100,
#                 nMax = 5e5,
#                 nInit = 10000,
#                 echoIt = FALSE)
# ================================================================
# Argumentos
# ----------
# mu, tau, alpha, beta: parâmetros da posteriori Normal-Gama
# paramBounds: lista de limitantes inferior e superior do espaço
#              paramétrico. Em caso de limitantes infinitos, definir
#              um valor suficientemente grande, mas limitado, para
#              evitar problemas com o otimizador.
# equalNHConstr: expressões das restrições de igualdade na Hipótese
#                Nula; deve haver pelo menos uma restrição.
# inequalNHConstr: expressões das restrições de desigualdade na Hipótese
#                  Nula.
# equalDomainConstr: restrições de igualdade sob o dominio da posteriori.
# inequalDomainConstr: restrições de desigualdade sobre o dominio da
#                      posteriori.
# initialPoint: ponto inicial para a otimização.
# impSampFunction: função de amostragem e densidade para o importance
#                  sampling, definida pelo usuário.
# alphacut: valor de corte (i.e. valor alpha que para Ev > alpha,
#           aceita-se H0 e para Ev < alpha, rejeita-se H0.
# rStar: risco (perda) máximo que se pode sofrer ao tomar a decisão ótima
# abc: vetor de parâmetros das funções de perda, caso o usuário queira
#      informá-los diretamente.
# custo: custo (unitário) de cada observação realizada.
# epsilon: critério de parada para as amostragens simuladas no processo
#
# parâmetros para execução do fbst: ver descrição e comentários
#                                   no arquivo fbst.r
# ================================================================
# Retorno
# -------
# integral: Evidência A FAVOR da hipótese nula para a posteriori avaliada.
# error: medida do erro cometido no cálculo da Evidência
# sample.size: quantidade de itens a serem amostrados
# riscos: Riscos Totais (incluído o custo de amostragem) de tomada da
#         decisão imediatamente e após k = 1, 2, ..., sample.size+1
#         observações serem amostradas
#
# Demais valores: ver comentários sobre a função evidence
#
# ================================================================
```

```
# Observação
# ----------
# Para não haver ambigüidade de notação da parametrização da posteriori
# Normal-Gama(mu, tau, alpha, beta), considera-se:
#
# * Os dados com distribuição (verossimilhança) Normal(m, v)
#
# * A precisão p (= 1/v) com distribuição (priori/posteriori)
#   Gama(alpha, beta)
#
# * A média m com distribuição (priori/posteriori) Normal(mu, 1/(tau*p))
#
# ******************************************************************
# ******************************************************************

sfbst.normgama <- function (mu,
                            tau,
                            alpha,
                            beta,
                            paramBounds,
                            equalNHConstr,
                            inequalNHConstr = NULL,
                            equalDomainConstr = NULL,
                            inequalDomainConstr = NULL,
                            initialPoint,
                            impSampFunction,
                            alphacut,
                            rstar,
                            abc = NULL,
                            custo,
                            epsilon = 0.0001,
                            ...) {

if (is.null(abc)) { a <- rstar/alphacut } else { a <- abc[1] }
if (is.null(abc)) { b <- 0 } else { b <- abc[2] }
if (is.null(abc)) { c <- rstar/(1-alphacut) } else { c <- abc[3] }

robarra <- vector()
robarra[1] <- NULL

# Construindo a densidade (na verdade o seu log) da posteriori a ser
# avaliada (considerando, conforme a notação da observação mais acima,
# x1 = m e x2 = p)

fposteriori <- paste("(",alpha - 0.5,"* log(x2)) - (",beta,
                     " * x2) - (",tau,"* x2 * ((x1 -",mu,")^2)/2)")

# Executando a "zerésima" iteração do processo

evbarra <- fbst(fposteriori, paramBounds, equalNHConstr, inequalNHConstr,
                equalDomainConstr, inequalDomainConstr, initialPoint,
                impSampFunction = impSampFunction, ...)

ev0 <- 1 - evbarra$integral
robarra0 <- min(a*ev0 , b+(c*(1-ev0)))
m0 <- robarra0
riscos <- vector()
riscos[1] <- robarra0
names(riscos)[1] <- 0

k <- 0
```

```
while (1)
  {
  k <- k+1
  theta.i <- vector()
  ro <- vector()
  ro[1] <- NULL
  i <- 0
  robarra[i] <- 0
  while(1)
    {
    i <- i+1
    prec.i <- rgamma(1, alpha, scale=beta)        # gera um vetor
    media.i <- rnorm(1, mu, sqrt(1/(tau*prec.i))) # (media.i, prec.i)
                                                  # com distr. Normal-Gama
                                                  # (mu, tau, alpha, beta)

    xsample <- rnorm(k, media.i, sqrt(1/prec.i))  # gera k observações com
                                                  # distribuição Normal
                                                  # (media.i, 1/prec.i)

# atualizando o log da densidade posteriori em função das k observações
# geradas

    alpha.i <- alpha + (k/2)
    beta.i <- beta + (((k*tau)*((mean(xsample)-mu)^2))/(2*(k+tau)))
    if (k > 1) beta.i <- beta.i + ((k-1)*var(xsample)/2)
    mu.i <- ((mu*tau)+sum(xsample))/(tau + k)
    tau.i <- tau + k

    posteriori.i <- paste("(",alpha.i - 0.5,"* log(x2)) - (",beta.i,
                          " * x2) - (",tau.i,"* x2 * ((x1 -",
                          mu.i,")^2)/2)")

    evbarra.i <- fbst(posteriori.i, paramBounds, equalNHConstr,
                      inequalNHConstr, equalDomainConstr,
                      inequalDomainConstr, initialPoint,
                      impSampFunction = impSampFunction, ...)

    ev.i <- 1 - evbarra.i$integral
    ro[i] <- min(a*ev.i , b+(c*(1-ev.i)))
    robarra[i] <- mean(ro)
    difro <- 0
    if (i > 1) difro <- abs(robarra[i] - robarra[i-1])/robarra[i-1]
    if ((i > 1) && (difro <= epsilon))
      {
      robarra.k <- robarra[i]
      riscos[k+1] <- robarra.k + (k*custo)
      names(riscos)[k+1] <- k
      break
      }
    }

  if ((robarra.k + (k*custo)) >= m0) break
  m0 <- min(m0 , robarra.k + (k*custo))
  }
evbarra[['sample.size']] <- k-1
evbarra$riscos <- riscos
evbarra$integral <- 1 - evbarra$integral
return(evbarra)
}
#########################################################################
```

A.4. Biblioteca sfbstapp.R

Observação: Essa e as demais bibliotecas se encontram disponíveis para *download* na página http://www.chancedegol.com.br/sfbst/sfbst.htm .

```
# ====================================================================
# Arquivo: sfbstapp.r
# ====================================================================
# Autor: Marcelo Leme de Arruda
# ====================================================================
#
# ====================================================================
# Modulo: Rotinas para aplicação do FBST Seqüencial
# a testes específicos.
#
# Last update of any of the component of this module:
#
# Nov 09, 2011.
#
######################################################################

source("fbst.r")

# ====================================================================
#
# Funcao: veross
#
# ====================================================================
# Descricao
# ---------
# Calcula a função de verossimilhança de uma amostra (X,Y) com
# distribuição Holgate (theta1,theta2,theta3)
#
# ====================================================================
# Uso
# ---
# veross(X,Y,theta1,theta2,theta3)
#
# ====================================================================
# Argumentos
# ----------
# X e Y: amostra cuja verossimilhança se quer calcular
# theta1,theta2,theta3: parâmetros da Holgate
#
# Observação: utiliza-se a aqui a notação segundo a qual
# E[X] = theta1, E[Y] = theta2 e Cov(X,Y) = theta3.
#
# ********************************************************************
# ********************************************************************
veross <- function (X,Y,theta1,theta2,theta3) {

n <- length(X)
if (theta3 > 0)
  {
  v <- 1
  for (i in 1:n)
    {
    s <- 0
    m <- min(X[i],Y[i])
    for (j in 0:m)
```

```
      {
      a <- ((theta1-theta3)^(X[i]-j))/factorial(X[i]-j)
      a <- a*((theta2-theta3)^(Y[i]-j))/factorial(Y[i]-j)
      a <- a*(theta3^j)/factorial(j)
      s <- s+a
      }
    s <- s*exp(-(theta1+theta2-theta3))
    v <- v*s
    }
  }
if (theta3 == 0)
  {
  v <- 1
  for (i in 1:n)
    {
    a <- exp(-theta1)*(theta1^X[i])/factorial(X[i])
    a <- a*exp(-theta2)*(theta2^Y[i])/factorial(Y[i])
    v <- v*a
    }
  }
veross <- v
}
# ===================================================================
#
# Função: evid.holgate
#
# ===================================================================
# Descrição
# ---------
# Função que calcula a evidência A FAVOR da hipótese de
# independência (contra a hipótese de dependência sob a
# distribuição bivariada de Holgate) de uma amostra.
#
# ===================================================================
# Uso
# ---
# evid.holgate (X,
#               Y,
#               alpha1,
#               beta1,
#               alpha2,
#               beta2,
#               N)
# ===================================================================
# Argumentos
# ----------
# X e Y: vetores previamente amostrados, cuja evidência se quer calcular.
# alpha1,beta1,alpha2,beta2: parâmetros das distribuições de importância
#                            Gama(alpha1,beta1) e Gama(alpha2,beta2).
# N: número de simulações a serem realizadas no Importance Sampling.
#
# *******************************************************************
# *******************************************************************

evid.holgate <- function (X,
                          Y,
                          alpha1,
                          beta1,
                          alpha2,
                          beta2,
                          N = 10000) {
```

```
sumzstar <- 0
sumz <- 0
k <- 0
histp <- vector()
while (k < N)
  {
  k <- k+1
  theta1 <- rgamma(1,shape=alpha1,rate=beta1)
  theta2 <- rgamma(1,shape=alpha2,rate=beta2)
  theta3 <- runif(1,0,min(theta1,theta2))
  gtheta <- dgamma(theta1,shape=alpha1,rate=beta1)*
            dgamma(theta2,shape=alpha2,rate=beta2)/min(theta1,theta2)
  Z <- veross(X,Y,theta1,theta2,theta3)
  lstar <- veross(X,Y,mean(X),mean(Y),0)
  Zstar <- Z
  if (Z < lstar) Zstar <- 0
  Z <- Z/gtheta
  Zstar <- Zstar/gtheta
  sumzstar <- sumzstar+Zstar
  sumz <- sumz + Z
  p <- sumzstar/sumz
  }

evid.holgate <- 1-p
}

# ====================================================================
#
# Função: indep.holgate
#
# ====================================================================
# Descrição
# ---------
# Função que executa o FBST Seqüencial para testar a hipótese
# de independência de duas variáveis com disptribuição de
# Poisson contra a hipótese de dependência sob a distribuição
# bivariada de Holgate.
#
# ====================================================================
# Uso
# ---
# indep.holgate (X,
#                Y,
#                alphacut,
#                rstar,
#                abc = NULL,
#                custo,
#                N = 10000,
#                epsilon = 0.0001)
# ====================================================================
# Argumentos
# ----------
# X e Y: vetores previamente amostrados, cuja independência se
#        quer testar.
# alphacut: valor de corte (i.e. valor alpha que para Ev > alpha,
#           aceita-se H0 e para Ev < alpha, rejeita-se H0.
# rStar: risco (perda) máximo que se pode sofrer ao tomar a decisão ótima
# abc: vetor de parâmetros das funções de perda, caso o usuário queira
#      informá-los diretamente.
# custo: custo (unitário) de cada observação realizada.
```

```
# N: número de simulações a serem realizadas no Importance Sampling
#    do cálculo da evidência.
# epsilon: critério de parada para as amostragens simuladas no processo.
#
# parâmetros para execução do fbst: ver descrição e comentários
#                                   no arquivo fbst.r
# ================================================================
# Retorno
# -------
# integral: Evidencia A FAVOR da hipotese nula para a amostra avaliada.
# sample.size: quantidade de itens a serem amostrados
# riscos: Riscos Totais (incluído o custo de amostragem) de tomada da
#         decisão imediatamente e após k = 1, 2, ..., sample.size+1
#         observações serem amostradas
#
# ****************************************************************
# ****************************************************************

indep.holgate <- function (X,
                           Y,
                           alphacut,
                           rstar,
                           abc = NULL,
                           custo,
                           N = 10000,
                           epsilon = 0.0001) {

if (is.null(abc)) { a <- rstar/alphacut } else { a <- abc[1] }
if (is.null(abc)) { b <- 0 } else { b <- abc[2] }
if (is.null(abc)) { c <- rstar/(1-alphacut) } else { c <- abc[3] }

alpha1 <- sum(X)
beta1 <- length(X)
alpha2 <- sum(Y)
beta2 <- length(Y)

robarra <- vector()
robarra[1] <- NULL

# Executando a "zerésima" iteração do processo

ev0 <- evid.holgate(X,Y,alpha1,beta1,alpha2,beta2,N)
evid <- list(integral = ev0,
             sample.size = 0)

robarra0 <- min(a*ev0 , b+(c*(1-ev0)))
m0 <- robarra0
riscos <- vector()
riscos[1] <- robarra0
names(riscos)[1] <- 0

k <- 0
while (1)
  {
  k <- k+1
  ro <- vector()
  ro[1] <- NULL
  i <- 0
  robarra[i] <- 0
  while(1)
    {
```

```
    i <- i+1

# gera um vetor (theta1,theta2,theta3) com distribuições
# Gama(alpha1,beta1), Gama(alpha2,beta2) e Uniforme(0,min(theta1,theta2)),
# respectivamente.

    theta1 <- rgamma(1,shape=alpha1,rate=beta1)
    theta2 <- rgamma(1,shape=alpha2,rate=beta2)
    theta3 <- runif(1,0,min(theta1,theta2))

# gera k observações (P1sim e P2sim) com distribuição
# Holgate(theta1,theta2,theta3)

    P12sim <- rpois(k,theta3)
    P1sim <- rpois(k,theta1-theta3)+P12sim
    P2sim <- rpois(k,theta2-theta3)+P12sim
    Xsim <- c(X,P1sim)
    Ysim <- c(Y,P2sim)
    a1 <- sum(Xsim)
    a2 <- sum(Ysim)
    b1 <- length(Xsim)
    b2 <- length(Ysim)

    ev.i <- evid.holgate(Xsim,Ysim,a1,b1,a2,b2,N)

    ro[i] <- min(a*ev.i , b+(c*(1-ev.i)))
    robarra[i] <- mean(ro)

    difro <- 0
    if (i > 1) difro <- abs(robarra[i] - robarra[i-1])/robarra[i-1]
    if ((i > 1) && (difro <= epsilon))
      {
      robarra.k <- robarra[i]
      riscos[k+1] <- robarra.k + (k*custo)
      names(riscos)[k+1] <- k
      break
      }
    }
  if ((robarra.k + (k*custo)) >= m0) break
  m0 <- min(m0 , robarra.k + (k*custo))
  }
evid[['thetaStar']] <- vector()
evid[['thetaStar']][1] <- mean(X)
evid[['thetaStar']][2] <- mean(Y)
evid[['thetaStar']][3] <- 0
evid[['sample.size']] <- k-1
evid[['riscos']] <- riscos
return(evid)
}

#############################################################################
# ===========================================================
#
# Função: behrens.fisher
#
# ===========================================================
# Descrição
# ---------
# Função que executa o FBST Seqüencial para testar a hipótese
# de igualdade entre as médias de duas disribuições normais
# independentes.
```

```
#
# ====================================================================
# Uso
# ---
# behrens.fisher (mu1,
#                 tau1,
#                 alpha1,
#                 beta1,
#                 mu2,
#                 tau2,
#                 alpha2,
#                 beta2,
#                 initialPoint,
#                 impSampFunction,
#                 alphacut,
#                 rstar,
#                 abc = NULL,
#                 custo,
#                 epsilon = 0.0001,
#
#                 indicatorMethod = FALSE,
#                 error = 0.01,
#                 gamma = 0.99,
#                 nDiv = 10,
#                 step = 100,
#                 nMax = 5e5,
#                 nInit = 10000,
#                 echoIt = FALSE)
# ====================================================================
# Argumentos
# ----------
# mu1, tau1, alpha1, beta1: parâmetros da posteriori Normal-Gama para
#                           a média e a precisão da variável X
# mu2, tau2, alpha2, beta2: parâmetros da posteriori Normal-Gama para
#                           o média e a precisão da variável Y
# paramBounds: lista de limitantes inferior e superior do espaço
#              paramétrico. Em caso de limitantes infinitos, definir
#              um valor suficientemente grande, mas limitado, para
#              evitar problemas com o otimizador.
# initialPoint: ponto inicial sob a hipótese nula para a otimização.
# impSampFunction: função de amostragem e densidade para o importance
#                  sampling, definida pelo usuário.
# alphacut: valor de corte (i.e. valor alpha que para Ev > alpha,
#           aceita-se H0 e para Ev < alpha, rejeita-se H0.
# rStar: risco (perda) máximo que se pode sofrer ao tomar a decisão ótima
# abc: vetor de parâmetros das funções de perda, caso o usuário queira
#      informá-los diretamente.
# custo: custo (unitário) de cada observação realizada.
# epsilon: critério de parada para as amostragens simuladas no processo
#
# parâmetros para execução do fbst: ver descrição e comentários
#                                   no arquivo fbst.r
# ====================================================================
# Retorno
# -------
# integral: Evidencia A FAVOR da hipotese nula para a posteriori avaliada.
# error: medida do erro cometido no cálculo da Evidência
# sample.size: quantidade de itens a serem amostrados
# riscos: Riscos Totais (incluído o custo de amostragem) de tomada da
#         decisão imediatamente e após k = 1, 2, ..., sample.size+1
#         observações serem amostradas
```

```
#
# Demais valores: ver comentarios sobre a função evidence
#
# ================================================================
# Observação
# ----------
# Para não haver ambigüidade de notação da parametrização da posteriori
# Normal-Gama(mu,tau,alpha,beta), considera-se:
#
# * Os dados com distribuição (verossimilhança) Normal(m,v)
#
# * A precisão p (= 1/v) com distribuição (priori/posteriori)
#   Gama(alpha,beta)
#
# * A média m com distribuição (priori/posteriori) Normal(mu, 1/(tau*p))
#
# ****************************************************************
# ****************************************************************

behrens.fisher <- function (mu1,
                            tau1,
                            alpha1,
                            beta1,
                            mu2,
                            tau2,
                            alpha2,
                            beta2,
                            paramBounds,
                            initialPoint,
                            impSampFunction,
                            alphacut,
                            rstar,
                            abc = NULL,
                            custo,
                            epsilon = 0.0001,
                            ...) {

if (is.null(abc)) { a <- rstar/alphacut } else { a <- abc[1] }
if (is.null(abc)) { b <- 0 } else { b <- abc[2] }
if (is.null(abc)) { c <- rstar/(1-alphacut) } else { c <- abc[3] }

equalNHConstr <- "- x1 + x3"
inequalNHConstr <- NULL
equalDomainConstr <- NULL
inequalDomainConstr <- NULL

robarra <- vector()
robarra[1] <- NULL

# Construindo a densidade (na verdade o seu log) da posteriori a ser
# avaliada (considerando, conforme a notação da observação mais acima,
# x1 = m e x2 = p, ambos relativos à variável X e x3 = m e x4 = p, ambos
# relativos á variável Y)

fposteriori <- paste("(",alpha1 - 0.5,"* log(x2)) - (",beta1," * x2) - (",
                     tau1,"* x2 * ((x1 -",mu1,")^2)/2) + (",
                     alpha2 - 0.5,"* log(x4)) - (",beta2," * x4) - (",
                     tau2,"* x4 * ((x3 -",mu2,")^2)/2)")


# Executando a "zerésima" iteração do processo
```

```
evbarra <- fbst(fposteriori, paramBounds, equalNHConstr, inequalNHConstr,
                equalDomainConstr, inequalDomainConstr, initialPoint,
                impSampFunction = impSampFunction, ...)

ev0 <- 1 - evbarra$integral
robarra0 <- min(a*ev0 , b+(c*(1-ev0)))
m0 <- robarra0
riscos <- vector()
riscos[1] <- robarra0
names(riscos)[1] <- 0

k <- 0
while (1)
  {
  k <- k+1
  theta.i <- vector()
  ro <- vector()
  ro[1] <- NULL
  i <- 0
  robarra[i] <- 0
  while(1)
    {
    i <- i+1

# gerando um vetor (media1.i, prec1.i) com distrib. Normal-Gama (mu1, tau1,
# alpha1, beta1) e depois k observações com distribuição Normal(media1.i,
# 1/prec1.i)

    prec1.i <- rgamma(1, alpha1, scale=beta1)
    media1.i <- rnorm(1, mu1, sqrt(1/(tau1*prec1.i)))

    xsample <- rnorm(k, media1.i, sqrt(1/prec1.i))

# gerando um vetor (media2.i, prec2.i) com distrib. Normal-Gama (mu2, tau2,
# alpha2, beta2) e depois k observações com distribuição Normal(media2.i,
# 1/prec2.i)

    prec2.i <- rgamma(1, alpha2, scale=beta2)
    media2.i <- rnorm(1, mu2, sqrt(1/(tau2*prec2.i)))

    ysample <- rnorm(k, media2.i, sqrt(1/prec2.i))

# atualizando o log da densidade posteriori em função das k observações
# geradas

    alpha1.i <- alpha1 + (k/2)
    beta1.i <- beta1 + (((k*tau1)*((mean(xsample)-mu1)^2))/(2*(k+tau1)))
    if (k > 1) beta1.i <- beta1.i + ((k-1)*var(xsample)/2)
    mu1.i <- ((mu1*tau1)+sum(xsample))/(tau1 + k)
    tau1.i <- tau1 + k

    alpha2.i <- alpha2 + (k/2)
    beta2.i <- beta2 + (((k*tau2)*((mean(ysample)-mu2)^2))/(2*(k+tau2)))
    if (k > 1) beta2.i <- beta2.i + ((k-1)*var(ysample)/2)
    mu2.i <- ((mu2*tau2)+sum(ysample))/(tau2 + k)
    tau2.i <- tau2 + k

    posteriori.i <- paste("(",alpha1.i - 0.5,"* log(x2)) - (",beta1.i,
                          " * x2) - (",tau1.i,"* x2 * ((x1 -",
                          mu1.i,")^2)/2)+(",
```

```
                        alpha2.i - 0.5,"* log(x4)) - (",beta2.i,
                        " * x4) - (",tau2.i,"* x4 * ((x3 -",
                        mu2.i,")^2)/2)")

    evbarra.i <- fbst(posteriori.i, paramBounds, equalNHConstr,
                      inequalNHConstr, equalDomainConstr,
                      inequalDomainConstr, initialPoint,
                      impSampFunction = impSampFunction, ...)

    ev.i <- 1 - evbarra.i$integral
    ro[i] <- min(a*ev.i , b+(c*(1-ev.i)))
    robarra[i] <- mean(ro)
    difro <- 0
    if (i > 1) difro <- abs(robarra[i] - robarra[i-1])/robarra[i-1]
    if ((i > 1) && (difro <= epsilon))
      {
      robarra.k <- robarra[i]
      riscos[k+1] <- robarra.k + (k*custo)
      names(riscos)[k+1] <- k
      break
      }
    }

  if ((robarra.k + (k*custo)) >= m0) break
  m0 <- min(m0 , robarra.k + (k*custo))
  }
evbarra[['sample.size']] <- k-1
evbarra$riscos <- riscos
evbarra$integral <- 1 - evbarra$integral
return(evbarra)
}

#############################################################################
```