Departamento de Estatística - Universidade de Brasília

GUILHERME HENRIQUE COELHO MENDES

MODELAGEM DE SEQUÊNCIAS DE DNA POR MEIO DE CADEIAS DE ORDEM VARIÁVEL

Brasília
2/2018

GUILHERME HENRIQUE COELHO MENDES

MODELAGEM DE SEQUÊNCIAS DE DNA POR MEIO DE CADEIAS DE ORDEM VARIÁVEL

Projeto apresentado para obtenção do título de Bacharel em Estatística.

Departamento de Estatística - Universidade de Brasília

Orientador: Prof. Dr. Lucas Moreira
Coorientador: Prof. Dr. Joanlise Marco de Leon Andrade

Brasília
2/2018

Sumário

Lista de tabelas

Lista de ilustrações

Resumo

O objetivo geral deste trabalho foi estudar as Cadeias de Ordem Variável, analisar e avaliar os estimadores do Algoritmo Contexto, a Versão Modificada do Algoritmo Contexto e do BIC e ainda aplicar a classe de modelos estudada em dados que compõe sua estrutura por cadeias de DNA, no caso deste trabalho as cadeias de DNA são representadas por doenças cromossomicas. Foi realizado um estudo computacional em dados simulados para estudarmos os estimadores e suas respectivas constantes de poda. Na parte da aplicação, modelamos os dados sobre doenças cromossomicas, encontramos uma árvore mais identificada entre 30 doenças retiradas da página `www.ncbi.nlm.nih.gov`, com o estimador BIC que foi o estimador que melhor obteve resultados.

Palavras-chave: Cadeias de Ordem Variável, Algoritmo Contexto, a Versão Modificada do Algoritmo Contexto, BIC, modelagem.

Introdução

Neste trabalho estudamos as Cadeias de Ordem Variável; avaliamos os estimadores do Algoritmo Contexto, da Versão Modificada do Algoritmo Contexto e do BIC; e realizamos uma aplicação modelando sequências de DNA que contém informações relacionadas a doenças genéticas e cromossomicas. Executamos as modelagem dessas sequências e comparamos algumas características entre elas. Isso foi implementado obtendo uma sequências de símbolos finitos de um lado da dupla fita de DNA. Essa sequência de DNA tem como estrutura as bases nitrogenadas que formam essa fita. As *Cadeias de Ordem Variável* foram utilizadas como classe de modelo neste trabalho.

As Cadeias de Ordem Variável foram introduzidas por Jorma Rissanen em 1983 no artigo *A universal data compression system*, onde apresentou seu modelo que o denominou como *Finitely Generated Source* com intuito de modelar sequências de dados. Essa família de cadeias estocásticas foi popularizada por P. Bühlmann e A. J. Wyner em 1999 no artigo *Variable Length Markov Chains* onde foram chamadas por Bühlmann e Wyner de "Cadeias de Markov de Alcance Variável", cuja sigla em inglês é VLMC. As cadeias de ordem variável são uma classe de modelos semelhante às cadeias markovianas com ordem k. De acordo com Bühlmann e Winder (1999) as cadeias de ordem variável são mais generalizantes e ricas, pois suas ordens variam conforme uma função do passado, o que permite uma economia no número de parâmetros utilizados no modelo e o seu emprego em casos não-Markovianos.

Rissanen denominou como *Contexto* a porção do passado suficiente para predizer um próximo símbolo. Ou seja, o contexto é a parte que influencia na probabilidade de transição para o próximo estado. O tamanho dessa porção relevante do passado é função do próprio passado e nenhum contexto pode ser representado por um sufixo de outro contexto. Assim podemos representar o conjunto de contextos por uma *árvore probabilistica de contextos*.

Em seu artigo, Rissanen também implementou o *Algoritmo Contexto* para estimar as árvores de contexto de uma cadeia de ordem variável. O princípio desse algoritmo se dava do seguinte modo, ele estimava a árvore, de forma que, um galho da árvore, representado por um contexto, só seria podado se a perda de informação presente nele não fosse significativa e não influenciasse na construção do modelo. A decisão de podar os galhos da árvore de contextos baseia-se em uma função ganho que envolve a distância de *Kullback-Leibler*.

As Cadeias de Ordem Variável formam uma classe de modelos que tem aplicações em diversas áreas, como genética, linguística, meteorologia, etc. Esses modelos podem ser

utilizados para descrever diversos processos estocásticos e são eficientes em várias situações, podendo ser considerados mais eficientes do que os modelos de Markov. As Cadeias de Ordem Variável são mais flexíveis e econômicas, pois consideram também as dependências estruturais do processo. Enquanto os parâmetros uma Cadeia de Markov aumentam de forma exponencial, conforme sua ordem aumenta, as correspondentes Cadeias de Ordem Variável tem um número de parâmetros significativamente menor. Ressaltmos que Cadeias de Ordem Variável se estendem a processos não-Markovianos, isto é, quando a árvore de contextos é não limitada.

Rissanen, apresentou também uma prova da consistência (fraca) do algoritmo no caso de uma cadeia fixa, ou seja, para a situação onde a árvore é limitada. Bühlman e Wyner (1999) também provaram a consistência fraca do Algoritmo Contexto proposto por Rissanen para cadeias finitas, ou seja, para o caso limitado, permitindo que a profundidade da árvore de contextos cresça de acordo com o tamanho da amostra. Ferrari and Wyner (2003) provaram a consistência fraca do Algoritmo Contexto para o caso em que a árvore é ilimitada. Csiszár e Tálata (2006) observando que mesmo sendo consistente, o algoritmo demorava muito para convergir, propuseram um algoritmo baseado no critério BIC (Bayesian Information Criterion) que era mais eficiente para a estimação. Mais tarde, Galves e Leonardi (2008) elaboraram uma versão modificada do Algoritmo Contexto com uma diferente função ganho considerando as diferenças entre sucessivas probabilidades de transição empíricas. Para comodidade do leitor, esses algoritmos serão apresentados e estudados no capítulo 1.

Este trabalho está organizado da seguinte forma: no Capítulo 1, apresentamos notações e conceitos básicos que serão necessários para compreensão do trabalho. Avaliamos o desempenho dos estimadores através de estudos computacionais no Capítulo 2. No Capítulo 3, apresentamos a abordagem, o cenário da aplicação em dados reais, os resultados e discussões dos dados codificados em sequências de símbolos no alfabeto $\mathcal{A} = \{0, 1, 2, 3\}$ aplicados pelo estimador BIC. O Capítulo 4 ficou reservado para as Considerações Finais deste trabalho.

1 Definições e Notações Básicas

Neste capítulo, apresentamos definições e notações básicas relativa às Cadeias de Ordem Variável e, para melhor compreensão do leitor, apresentamos alguns exemplos.

1.1 Notações e Conceitos

Antes de definirmos formalmente uma Cadeia de Ordem Variável realizamos uma revisão dos conceitos básicos relacionados a processos estocásticos. Consideramos primeiramente um alfabeto finito $\mathcal{A} = \{0, 1, 2, ..., N-1\}$, sua cardinalidade, ou seja, seu tamanho, é dado por $|\mathcal{A}| \in \mathbb{N}, N > 0$. Em seguida, tomamos dois inteiros m e n, onde $m \leq n$. A sequência de símbolos $a_m a_{m+1}...a_n$ do alfabeto $\mathcal{A}$ é denotada por a_m^n e seu comprimento definido por $l(a_m^n) = n - m + 1$. Logo, se a sequência de símbolos for $a_{-3}a_{-2}a_{-1}$, representamos ela por a_1^3 e seu comprimento é $l(a_1^3) = 3 - 1 + 1 = 3$. A sequência vazia é denotada por $\emptyset$, com comprimento $l(\emptyset) = 0$. Se $n < m$, temos que $a_m^n = \emptyset$.

Representamos por $\mathcal{A}^j$ o conjunto de todas as sequências com mesmo tamanho j com símbolos pertencentes ao alfabeto $\mathcal{A} \in \mathbb{N}$. O conjunto das sequências de símbolos a_m^n é representado por $\mathcal{A}_m^n$. Adicionalmente, denotamos o conjunto de todas as sequências semi-infinitas por

$$\mathcal{A}^\infty = \mathcal{A}_{-\infty}^{-1} = \mathcal{A}^{...,-2,-1}$$

e o conjunto de todas as sequências de símbolos de tamanho finito é denotado por

$$\mathcal{A}^* = \bigcup_{j=0}^{\infty} \mathcal{A}_{-j}^{-1},$$

sendo que $\mathcal{A}^0$ representa o conjunto das sequências vazias $\emptyset$.

Admitimos duas sequências u e v, onde $u \in \mathcal{A}^* \bigcup \mathcal{A}^\infty$ e $v \in \mathcal{A}^*$. Denotamos por uv a sequência obtida pela concatenação entre u e v, com comprimento $l(u)+l(v)$. Por exemplo, se $u = u_1^m$ e $v = v_1^n$, com $m \geq 1$ e $n \geq 1$ inteiros, então a concatenação entre u e v é dada por $uv = u_1u_2...u_mv_1...v_n$. Caso $v = \emptyset$, temos $uv = u\emptyset = u$, sendo analogo para o caso $u = \emptyset$.

Considere as sequência $w \in \mathcal{A}^*$ e $u \in \mathcal{A}^* \bigcup \mathcal{A}^\infty$. Caso exista $v \in \mathcal{A}^* \bigcup \mathcal{A}^\infty$, com $l(v) \geq 1$, tal que $u = vw$, então w é um sufixo de u. Essa relação pode ser simbolizada por $w \prec u$. Se $w \prec u$ ou $w = u$, então $w \preceq u$. O maior sufixo de uma sequência w finita é denotado por $suf(w)$.

Um conjunto $\mathcal{T} \in \mathcal{A}^* \bigcup \mathcal{A}^\infty$ de sequências pode ser escrita como uma árvore probabilística se nenhuma dessas sequências pertencentes a $\mathcal{T}$ for sufixo de outra seqûencia pertencente $\mathcal{T}$. Chamamos essa propriedade de *propriedade do sufixo*. Os elementos da árvore $\mathcal{T}$ são chamados de folhas de $\mathcal{T}$ e um nó interno é um sufixo de uma folha.

Uma árvore $\mathcal{T}$ é dita completa se cada nó interno tem $|\mathcal{A}|$ descendentes, onde os descendentes de um nó interno são todas as sequências $as, a \in \mathcal{A}$. Se nenhuma sequência $s \in \mathcal{T}$ puder ser substituída por um sufixo de s sem violar a propriedade do sufixo, então dizemos que a árvore é irredutível. Essa noção foi introduzida em Csiszár e Talata (2006) e generaliza o conceito de árvore completa. Denotamos por $|\mathcal{T}|$ a cardinalidade de $\mathcal{T}$. A profundidade de uma árvore $\mathcal{T}$ é definida como

$$h(\mathcal{T}) = \max\{l(s), s \in \mathcal{T}\}. \tag{1.1}$$

Se $h(\mathcal{T}) < \infty$, dizemos que $\mathcal{T}$ é *limitada*. Caso contrario, dizemos que $\mathcal{T}$ é *ilimitada*. Dado um inteiro K, $\mathcal{T}|_K$ denota a árvore $\mathcal{T}$ truncada em K, ou seja,

$$\mathcal{T}|_K = \{s \in \mathcal{T} : l(s) \leq K\} \bigcup \{s \in \mathcal{A}^k : s \prec s' \text{ para algum } s' \in \mathcal{T}\}. \tag{1.2}$$

Consideramos o processo $X = \{X_t, t \in \mathbb{Z}\}$ estacionário e ergótico sobre o alfabeto $\mathcal{A} = \{0, 1, ..., N-1\}$. Assumimos que o processo X é compatível com a probabilidade de transição $p_X(.|.)$. Então:

$$p_X(a|w) = \mathbb{P}(X_0 = a | X_{-1} = w_{-1}, X_{-2} = w_{-2}, ...) \tag{1.3}$$

para todo $w \in \mathcal{A}_{-\infty}^{-1}$ e para todo $a \in \mathcal{A}$. Para $w \in \mathcal{A}_{-j}^{-1}$ a probabilidade estacionária do cilindro definida por essa sequência será denotada por

$$\mu_X(w) = \mathbb{P}(X_{-j}^{-1} = w). \tag{1.4}$$

As Cadeias de Ordem Variável possuem a propriedade de agurpar determinadas probabilidades de transição da cadeia, gerando, assim, uma característica de memória variável. Isso faz com que o número de passos anteriores necessários para a predição do elemento atual seja uma função do próprio passado, variando de símbolo para símbolo.

Considerando a hipótese de estacionariedade, a distribuição de probabilidade $\mathbb{P}$ de uma cadeia de ordem variável é determinada pelas suas respectivas probabilidades de transição $\mathbb{P}(X_0 = x_0 | t(x_{-\infty}^{-1}))$. Sendo assim, podemos representar o seu conjunto de parâmetros através de uma árvore, denominada como *árvore probabilística de contextos*. Os elementos dessa árvore são os contextos gerados pela função contexto associada as cadeias de ordem variável. Importante notar que, de acordo com a definição da função contexto, a propriedade do sufixo é satisfeita.

Portanto, podemos representar uma cadeia de ordem variável da seguinte forma:

1. No topo temos o primeiro nó, que é a raiz, que significa o estado presente;

2. Os galhos (também chamados de ramos ou descendentes) são os passados relevantes. Quanto mais a raiz está longe do nó, mais passos passados foram tomados e precisam ser observados;

3. Os galhos crescem de cima para baixo;

4. Cada nó tem no máximo $|\mathcal{A}|$ galhos, que é o tamanho do espaço de estados da cadeia;

5. Os contextos são os galhos que ligam o último nó à raiz, sendo que cada um é representado por um galho completo;

Exemplo 1.1: Representação da árvore de contextos *Considere o alfabeto e a árvore,*$A = \{0, 1, 2\}$ *e* $\mathcal{T} = \{000, 100, 10, 1, 2\}$*, representados respectivamente. Representamos graficamente a árvore* $\mathcal{T}$ *na Figura 1. O conjunto* $\mathcal{T}$ *satisfaz a propriedade do sufixo, portanto é de fato uma árvore. Em* $\mathcal{T}$*:*

- as sequências $000, 100, 10, 1, 2$ são folhas;
- $00, 0$ e $\emptyset$ são nós internos;
- $000, 100, 00, 10, 0, 1$ e $\emptyset$, são nós.

O nó interno 00 *possui apenas os descendentes* 000 *e* 100*, logo a árvore* $\mathcal{T}$ *não é completa. A árvore* $\mathcal{T}$ *truncada em 3 é dada por* $\mathcal{T}|_2 = \{00, 10, 1, 2\}$*. Além disso, a árvore* $\mathcal{T}$ *é limitada e irredutível.*

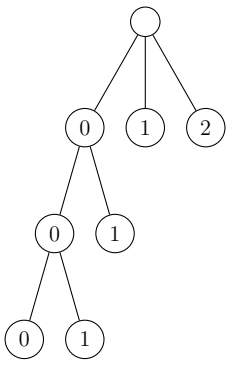


Figura 1 – Árvore de contextos do exemplo 1.1

As definições a seguir formalizam o conceito de uma cadeia de ordem variável e quando essa classe de modelos é compatível com um processo X:

Definição 1.1 *Dado um processo estacionário e ergódico* $X = \{X_t : t \in \mathbb{Z}\}$ *em um alfabeto finito* $\mathcal{A} = \{0, 1, ..., N-1\}$*. Considerando essa sequência* $s \in \mathcal{A}^*$*, escrevemos*

$$\mu_X(s)\begin{cases}\mathbb{P}(X_1^{l(s)}) = s, & \text{se } s \neq \emptyset;\\ 0, & \text{se } s = 0.\end{cases}$$

Dadas as sequências s e a, onde $s \in \mathcal{A}^* \bigcup \mathcal{A}^\infty$ e $a \in \mathcal{A}$, respectivamente, temos

$$p_X(a|s) = \begin{cases}\mathbb{P}(X_0 = a|X_{-l(s)}^{-1}) = s, & \text{se } s \neq \emptyset;\\ \mu_X(a), & \text{se } s = 0.\end{cases}$$

Definição 1.2 *Uma Cadeia de Ordem Variável em um alfabeto $\mathcal{A}$ um par ordenado $(\mathcal{T}, \bar{p})$ que satisfaz as condições:*

1. *$\mathcal{T}$ é uma árvore irredutível;*
2. *$\bar{p} = \{\bar{p}(.|s), s \in \mathcal{T}\}$ é uma família de probabilidades de transição sobre $\mathcal{A}$.*

Definição 1.3 *O processo X será compatível com a cadeia de ordem variável $(\mathcal{T}, \bar{p})$ caso satisfaça as condições:*

1. *$\mathcal{T}$ é a árvore de contextos do processo X;*
2. Para qualquer $s \in \mathcal{T}$ e $a \in \mathcal{A}$, $p_X(a|s) = \bar{p}(a|s)$.

Denotamos por $\mathcal{T}_X$ a cadeia de ordem variável do processo X.

1.2 Processos de Estimação de Árvores de Contextos

Seja $X_1, X_2, ..., X_n$ uma amostra aleatória do processo X. Dado um inteiro positivo d, o número de ocorrências da sequência s é denotado por $N_n(s, a)$, onde $s \in \bigcup_{j=0}^{d} \mathcal{A}^j$ seguida pelo símbolo $a \in \mathcal{A}$, ou seja,

$$N_n(s, a) = \sum_{i=d+1}^{n} \mathbb{I}(X_{i-l(s)}^{i-1} = s, X_i = a). \tag{1.5}$$

Seja $N_n(s)$ o número de ocorrências de s, ou seja, em toda a amostra é

$$N_n(s) = \sum_{a \in \mathcal{A}}^{n} N_n(s, a). \tag{1.6}$$

Se $s = \emptyset$, convencionamos $N_n(s, a) = \sum_{i=d+1}^{n} \mathbb{I}(X_{i-l(s)}^{i-1} = s$ e $N_n(s) = n - d$.

O conjunto de todas as sequências finitas $s = x_{-j}^0$, tal que $j \leq d$, isto é, $s \in \bigcup_{j=0}^{d} \mathcal{A}^j$, que aparecem ao menos uma vez na amostra será denotado por $\mathcal{V}_n$. Ou seja,

$$\mathcal{V}_n = \{s \in \bigcup_{j=0}^{d} \mathcal{A}^j : N_n(s) \geq 1\}. \tag{1.7}$$

A seguir, definimos quando uma árvore é factível. O objetivo dessa definição é, a partir da amostra, fazer uma seleção inicial das possíveis árvores de contexto de X.

Definição 1.4: *Uma árvore $\mathcal{T}$ é factível se satisfaz:*

1. *$s \in \mathcal{V}_n$ para todo $s \in \mathcal{T}$;*
2. *Cada sequência $s' \in \mathcal{V}_n$ é tal que $s' \preceq s$ ou $s \preceq s'$ para algum $s \in \mathcal{T}$.*

O conjunto de todas as árvores factíveis será denotado por $\mathcal{F}_n$. O objetivo é estimar a árvore de contextos $\mathcal{T}_X$ a partir de uma amostra de X. Para tanto, devemos escolher uma árvore factível que se aproxime de $\mathcal{T}_X$. Se $h(\mathcal{T}_X) < \infty$, então devemos escolher o inteiro d de modo que $h(\mathcal{T}_X) \leq d$ para que exista uma árvore factível que coincida com $\mathcal{T}_X$. Para estimar $\mathcal{T}_X$, não é necessário o conhecimento prévio da sua profundidade, portanto d pode ser uma função crescente de n.

Dada uma árvore $\mathcal{T}_X \subset \bigcup_{j=0}^{d} \mathcal{A}^j$ de um processo X, a probabilidade de ocorrer x_1^n pode ser escrita como:

$$\mathbb{P}(X_1^n = x_1^n) = \mathbb{P}(X_1^d = x_1^d)\, \Pi_{s\in\mathcal{T}}\, \Pi_{a\in\mathcal{A}}\; p_X(a|s)^{N_n(s,a)}. \tag{1.8}$$

Assumiremos $\mathbb{P}(X_1^d = x_1^d) = 1$. A máxima verossimilhança da amostra x_1^n é dada por

$$\mathrm{ML}_{\mathcal{T}}(x_1^n) = \Pi_{s\in\mathcal{T}}\; \Pi_{a\in\mathcal{A}}\; \hat{p}_n(a|s)^{N_n(s,a)}. \tag{1.9}$$

em que as probabilidades empíricas $\hat{p}_n(a|s)$ são dadas por

$$\hat{p}_n(a|s) = \begin{cases} \frac{N_n(s,a)}{N_n(s)}, & \text{se } N_n(s) > 0; \\ \frac{1}{|\mathcal{A}|}, & \text{se } N_n(s) = 0. \end{cases}$$

Para uma sequência $s \in \bigcup_{j=0}^{d} \mathcal{A}^j$, seja

$$\mathrm{ML}_s(x_1^n) = \Pi_{a\in\mathcal{A}}\; \hat{p}_n(a|s)^{N_n(s,a)}.$$

Portanto,

$$\mathrm{ML}_{\mathcal{T}}(x_1^n) = \Pi_{s\in\mathcal{T}}\; \mathrm{ML}_s(x_1^n).$$

Para todo elemento $a \in \mathcal{A}$ e para toda sequência finita s, a probabilidade de transição empírica é dada por

$$\hat{p}_Z(a|s)_n = \frac{N_n(sa) + 1}{N_n(s\cdot) + |\mathcal{A}|}. \tag{1.10}$$

Observe que a definição de $\hat{p}_Z(a|s)_n$ é conveniente pois é assintoticamente equivalente ao Estimador de Máxima Verossimilhançaa que é $\frac{N(sa)}{N(s\cdot)}$ e evita-se uma definição adicional

no caso $N(s\cdot) = 0$.

Exemplo 1.2: (Estimação das Probabilidades de Transição) *Considere X uma Cadeia de Ordem Variável tomando valores em um alfabeto $\mathcal{A} = \{0, 1\}$ e com árvore de contextos $\mathcal{T}_X = \{0, 01, 11\}$. Seja 111001011010111 uma amostra aleatória do processo X, com tamanho igual a 15.*

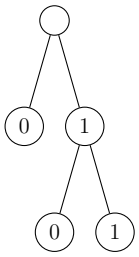


Figura 2 – Árvore de contextos do exemplo 1.2

Podemos ver que o número de ocorrências dos contextos $\omega = 0$, $v = 01$ e $u = 11$ foram dadas, respectivamente, por $N_{15}(\omega) = 5$, $N_{15}(v) = 4$ e $N_{15}(u) = 5$. Com intuito de estimar as probabilidades de transição deste processo é necessário determinar o número de ocorrências da concatenação entre cada contexto com o estado 0. Foram obtidas $N_{15}(00) = 1$, $N_{15}(010) = 2$ e $N_{15}(110) = 2$. As probabilidades de transição foram estimadas da seguinte forma

$$\hat{p}_X(0|0)_{15} = \frac{N_{15}(00) + 1}{N_{15}(\omega) + |\mathcal{A}|} = \frac{1+1}{5+2} = 0,28572 \approx 0.286$$

onde, a cardinalidade $|\mathcal{A}| = 2$.

As probabilidades de transição estimadas foram $\hat{p}_X(0|0)_{15} = 0,286$, $\hat{p}_X(0|01)_{15} = 0,5$ e $\hat{p}_X(0|11)_{15} = 0,429$.

Tabela 1 – Probabilidades de Transição da Árvore do exemplo 1.2

	α	
$\hat{p_Z}(a\|\omega)_{15}$	0	1
0	0,286	0,714
01	0,5	0,5
11	0,429	0,571

Para predição do próximo estado do processo , precisamos olhar para um período anterior, caso o estado observado seja 0 então não é mais necessário olhar para estados a mais no passado. Especificamente para o processo do Exemplo 2.2, a probabilidade do próximo estado ser 1 dado que no estado anterior foi observado 0 é de $\hat{p_Z}(1|0)_{15} = 0,714$, e $\hat{p_Z}(0|0)_{15} = 0,286$ para a probabilidade do próximo estado ser 0.

Caso o estado anterior observado seja 1, é preciso olhar para mais um período anterior na cadeia para predizermos o próximo estado. Então caso este estado seja 0 temos $\hat{p_Z}(1|01)_{15} = 0,5$ *é a probabilidade do próximo estado ser 1 dado que observamos 1 e 0 no passado. E* $\hat{p_Z}(1|11)_{15} = 0,572$ *é a probabilidade do próximo estado ser 1 dado que se observou 1 e 1 no passado.*

1.3 O Algoritmo Contexto

O Algoritmo Contexto proposto por Rissanen (1983) fundamenta-se, a princípio, na árvore completa, ou seja, na maior árvore factível. Logo após, mensuramos a discrepância entre as probabilidades empíricas de transição do maior sufixo de um contexto e seus descendentes. Se a discrepância for maior que um dado valor limiar, o contexto será mantido; caso contrário, será podado. Em outras palavras, caso a perda de informação presente em um galho não seja necessáriamente relevante, ele é podado. Esse processo ocorre até que até que não seja mais viável podar a árvore.

Rissanen utilizou a divergência de *Kullback-Leibler* para medir discrepância, ou seja, a diferença entre duas medidas de probabilidade em um alfabeto $\mathcal{A}$. Definimos esse método abaixo.

Definição 1.5: *A divergência de* ***Kullback-Leibler****, definida para medidas de probabilidade* P *e* Q *em um alfabeto* $\mathcal{A}$*, é dada por*

$$D(P;Q) = \sum_{a\in\mathcal{A}} P(a) log \frac{P(a)}{Q(a)} \tag{1.11}$$

Por convenção,

$$\begin{cases} P(a)log\frac{P(a)}{Q(a)} = 0, & \text{se } P(a) = 0; \\ P(a)log\frac{P(a)}{Q(a)} = +\infty, & \text{se } P(a) > Q(a) = 0. \end{cases}$$

Para uma sequência $s \in \mathcal{V}_n$, seja

$$\Lambda_n(s) = \sum_{b\in\mathcal{A}:bs\in\mathcal{V}_n} N_n(bs) D(\hat{p}_n(.|bs); \hat{p}_n(.|s)). \tag{1.12}$$

Denotamos o limiar utilizado no Algoritmo Contexto por δ_n, em que $(\delta_n)_{n\in\mathbb{N}}$ é uma sequência de números reais de modo que $\delta_n \to \infty$ e $\delta_n/n \to 0$ quando $n \to \infty$. Podemos descrever o Algoritmo Contexto em três passos:

- Passo 1: Seja $X_1, X_2, ..., X_n$ uma amostra aleatória do processo X e considere $\hat{\mathcal{T}}_0$ a maior árvore factível obtida a partir dessa amostra, isto é, a árvore $\hat{\mathcal{T}}_0$ que possui a maior cardinalidade $|\hat{\mathcal{T}}_0|$.

- Passo 2: Seja $S = \{suf(\omega) : \omega \in \hat{\mathcal{T}}_0\}$. Aplique a função $\Lambda_n(s)$ para todo $s \in S$, com $N_n(s) \geq 2$, tal que não exista $s' \in S$ e $s \prec s'$. Se $\Lambda_n(s) < \delta_n$, substitua por s todos os elementos $as \in \hat{\mathcal{T}}_0$, com $a \in \mathcal{A}$. Denotamos por $\hat{\mathcal{T}}_1$ a árvore obtida após o procedimento.

- Passo 3: Repita o Passo 2 para $\hat{\mathcal{T}}_i$ em vez de $\hat{\mathcal{T}}_{i-1}$, com $i = 1, 2, ...$, até que a árvore obtida após o procedimento seja idêntica à inicial. Denotamos essa árvore por $\hat{\mathcal{T}}_C(X_1^n)$.

Para uma amostra X_1^n , podemos também obter $\hat{\mathcal{T}}_C(X_1^n)$ a partir da função $C_s(X_1^n)$, definida para todo $s \in \mathcal{V}_n$ e dada por

$$C_s(X_1^n) = \begin{cases} 0, & \text{se } N_n(s) \leq 1 \text{ ou } l(s) = d \\ max\{\mathbb{I}(\Lambda_n(s) \geq \delta_n), max_{b\in\mathcal{A}} C_{bs}(X_1^n)\}, & \text{se } N_n(s) > 1 \text{ e } l(s) < d. \end{cases}$$

Definição 1.6 *O estimador $\hat{\mathcal{T}}_C(X_1^n)$ da árvore de contextos de X é o conjunto dado por*

$$\hat{\mathcal{T}}_C(X_1^n) = \{s \in \mathcal{V}_n : C_s(X_1^n) = 0 e C_{s'}(X_1^n) = 1 \text{ para todo } s' \prec s\}. \tag{1.13}$$

1.4 Uma Versão Modificada do Algoritmo Contexto

Uma modificação do Algoritmo Contexto de Rissanen (1983) foi proposta em Galves e Leonardi(2008). Nesta variação, tomamos a decisão de poda a partir das diferenças entre probabilidades condicionais empíricas. Então, seja $\Delta_n(s)$ o operador

$$\Delta_n(s) = \max_{a\in\mathcal{A}} |\hat{p}_n(a|s) - \hat{p}_n(a|suf(s))|, \forall s \in \bigcup_{k=1}^{\infty} \mathcal{A}^k. \tag{1.14}$$

Definição 1.7 *Para quaisquer $\delta > 0$ e $d < n$, o estimador da árvore de contextos $\hat{\mathcal{T}}_n^{\delta,d}$ é o conjunto que contém todas as sequências $s \in \bigcup_{k=1}^{d} \mathcal{A}^k$, tais que $\Delta_n(asuf(s)) > \delta$, para algum $a \in \mathcal{A}$, é $\Delta_n(us) \leq \delta$, para todo $u \in \bigcup_{k=1}^{d-l(s)} \mathcal{A}^k$.*
Se $\hat{\mathcal{T}}_n^{\delta,d} = \emptyset$, consideraremos $\hat{\mathcal{T}}_n^{\delta,d} = \{\emptyset\}$. No caso $l(s) = d$, definimos $\bigcup_{k=1}^{0} \mathcal{A}^k = \emptyset$.

Observe que as contantes $\delta > 0$ e $d < n$ são fundamentais para o estimador, pois, inicialmente consideramos a árvore completa, ou seja, a árvore de contextos maximal, assim cada sequência s candidata a contexto possui comprimento $l(s) = d$, ou seja, $s \in \mathcal{A}_{-d}^{-1}$. Em

seguida, o estimador reduz o comprimento das sequências s que não satisfazem o critério de poda, apresentado na Definição 2.1.8, tomando $suf(s)$, ou seja, o maior sufixo de s, como novo candidato a contexto. Este procedimento é repetido até que a condição de parada seja satisfeita para todas as sequências $s \in \hat{\mathcal{T}}_n^{\delta,d}$.

Agora analisando a consistência do estimador, para que a versão modificada do Algoritmo Contexto seja fortemente consistente, vamos supor que a árvore probabilística de contextos $(\mathcal{T}, p)$ satisfaz as condições das duas definições a seguir.

Definição 1.8 *Dizemos que uma árvore probabilística de contextos $(\mathcal{T}, p)$ é fortemente não nula, se satisfaz*

$$\inf_{a\in\mathcal{A}, s\in\mathcal{T}}\{p(a|s)\} > 0. \tag{1.15}$$

Definição 1.9 *Seja a sequência $\{\alpha_k\}_{k\geq 0}$, definida por*

$$\alpha_0 = \sum_{a\in\mathcal{A}} \inf_{s\in\mathcal{T}}\{p(a|s)\},$$
$$\alpha_k = \sum_{u\in\mathcal{A}^k} \inf_{s\in\mathcal{T}, u\prec s}\{p(a|s)\}. \tag{1.16}$$

Dizemos que uma árvore probabilística de contextos $(\mathcal{T}, p)$ é somável se possui sequência $\{\beta_k\}_{k\geq 0}$, $\beta_k = 1 - \alpha_k$, tal que

$$\beta = \sum_{k\in\mathbb{N}} \beta_k < 0. \tag{1.17}$$

Também é necessário restringir os valores de δ e d para que o estimador seja fortemente consistente. Com base nisso, dado um inteiro $k \geq 1$ e uma árvore probabilística de contextos $(\mathcal{T}, p)$, definimos

$$\mathcal{C}_k = \{u \in \mathcal{T}|_k : p(a|u) \neq p(a|suf(u)), \text{para alguma} a \in \mathcal{A}\},$$
$$D_k = \min_{u\in\mathcal{C}_k} \max_{a\in\mathcal{A}}\{p(a|u) - p(a|suf(u))\}.$$

Note que $D_k > 0$ para todo $k \geq 1$. Podemos enunciar agora o teorema que trata da convergência forte da versão modificada do Algoritmo Contexto introduzida em Galves e Leonardi (2008).

Teorema 1: *Seja $X_1, ..., X_n$ uma amostra aleatória de um processo $X = \{X_t : t \in \mathbb{Z}\}$ estacionário e ergódico, compatível com a árvore probabilística de contextos $(\mathcal{T}, p)$ fortemente não-nula e somável. Então, para qualquer inteiro $K, 0 < \delta < D_d$ e d satisfazendo*

$$d > \max_{u\notin\mathcal{T}, l(u)\leq K} \min\{k : \exists s \in \mathcal{C}_k, u \prec s\},$$

temos,

$$\hat{\mathcal{T}}_n^{\delta,d}|_K = \mathcal{T}|_K$$

quase certamente quando $n \to \infty$.

Observação: Para que o estimador seja consistente, é necessário tomar δ pequeno o suficiente. Isso significa que o estimador $\hat{\mathcal{T}}_n^{\delta,d}$ não é universal, porque os valores que δ pode assumir dependem do processo X que deu origem à amostra.

1.5 O Critério BIC

Considere $X_1, X_2, ..., X_n$ uma amostra do processo X. A seleção de uma árvore factível $\mathcal{T}_0 \subset \mathcal{F}_n$ que estime $\mathcal{T}_X$ deve avaliar e considerar a função de verossimilhança da amostra e a complexidade da árvore. A fim de estimarmos a árvore de contexto, temos como objetivo, escolher o valor de $\mathcal{T}_0$ de modo que a função de verossimilhança da amostra seja maximizada e comparativamente alta, dado que temos preferência por modelos menos complexos. Definimos o Critério de Informação Bayesiana (BIC) a seguir.

Definição 1.10 *Dada uma amostra* X_1^n*, o Critério de Informação Bayesiana (BIC) para uma árvore factível* $\mathcal{T}$ *é definida como*

$$BIC_{\mathcal{T}}(X_1^n) = -\log ML_{\mathcal{T}}(X_1^n) + c|\mathcal{T}| \log n, \tag{1.18}$$

em que $\boldsymbol{c}$ *é uma constante real positiva.*

Podemos ainda destacar uma particularidade do critério BIC que é sua constante de penalização c. Observamos que a penalização não é constante, ele altera de acordo com o tamanho da amostra. Em Czizár e Talata (2006) foi escolhido a constante $c = (|A| - 1)/2$ para a profundidade da árvore estimada, ou seja, $c|\mathcal{T}|$ representa metade do número de parâmetros livres do modelo quando a árvore $\mathcal{T}$ é completa.

Definição 1.11 *estimador BIC é dado por*

$$\hat{\mathcal{T}}_{BIC}(X_1^n) = arg \min_{\mathcal{T} \in \mathcal{F}_n} BIC_{\mathcal{T}}(X_1^n). \tag{1.19}$$

Na prática, é inviável estimar a árvore de contextos calculando o critério BIC para cada árvore factível. Entretanto, Czizár e Talata (2006) obtiveram um algoritmo eficiente para encontrar $\hat{\mathcal{T}}_{BIC}(X_1^n)$. Para a conveniência do leitor, iremos descrevê-lo a seguir. Definimos, recursivamente, para todo $s \in \mathcal{V}_n$, a função

$$V_s(x_1^n) = \begin{cases} max\{n^{-c}ML_s(x_1^n), \Pi_{b\in\mathcal{A}:bs\in\mathcal{V}_n} V_{bs}(x_1^n)\}, & \text{sel}(s) < d \\ n^{-c}ML_s(x_1^n), & \text{sel}(s) = d \end{cases}$$

e a indicadora

$$\mathcal{X}(x_1^n) = \begin{cases} \mathbb{I}\{\Pi_{b\in\mathcal{A}:bs\in\mathcal{V}_n} V_{bs}(x_1^n) > n^{-c}ML_s(x_1^n)\}, & \text{sel}(s) < d \\ 0, & \text{sel}(s) = d \end{cases}$$

Convencionamos que se $\{b \in \mathcal{A} : bs \in \mathcal{V}_n\} = \emptyset$, então $\mathcal{V}_s(x_1^n) = n^{-c}ML_s(x_1^n)$ e $\mathcal{X}_s(x_1^n) = 0$. Czizár e Talata (2006) demonstraram que

$$\hat{\mathcal{T}}_{BIC}(X_1^n) = \{s \in \mathcal{V}_n : \mathcal{X}_s(x_1^n) = 0\} \tag{1.20}$$

e

$$\mathcal{X}_s(x_1^n) = 1 \text{ para todo } s' \prec s \tag{1.21}$$

Czizár e Talata (2006) também provaram a consistência forte do estimador BIC para árvores de contexto finitas e infinitas. Enunciamos abaixo esse resultado.

Teorema 2: *Czizár e Talata (2006). Seja $X_1, ..., X_n$ uma amostra aleatória de um processo $X = \{X_t : t \in \mathbb{Z}\}$ ergódico e estacionário, compatível com uma árvore probabilística de contextos $(\mathcal{T}, p)$. No caso em que $h(\mathcal{T}) < \infty$, o estimador BIC, com $d = o(\log(n))$, satisfaz*

$$\hat{\mathcal{T}}_{BIC}(X_1^n) = \mathcal{T} \tag{1.22}$$

quase certamente quando $n \to \infty$. No caso ilimitado, para qualquer K natural, o estimador BIC satisfaz

$$\hat{\mathcal{T}}_{BIC}(X_1^n)|_K = \mathcal{T}|_K \tag{1.23}$$

quase certamente quando $n \to \infty$.

2 Estudos Computacionais

Neste capítulo, testamos e avaliamos o desempenho dos estimadores de árvores de contextos por meio de simulações. Consideramos o Algoritmo Contexto, a Versão Modificada do Algoritmo Contexto proposta por Galves e Leonardi (2008) e o critério BIC. Todos as definições foram descritas no Capítulo 1, das Definições e Notações Básicas.

Verificamos os desempenhos dos estimadores analisando diferentes valores previamente fixados para as respectivas constantes de poda de cada estimador. A constante C compõe o limiar δ_n para poda no Algoritmo Contexto, sendo que $\delta_n = C\log(n)$. Para o critério BIC, de uma árvore factível $\mathcal{T}$ para uma amostra X_1^n, teríamos $c|\mathcal{T}|$ no termo $c|\mathcal{T}|\log(n)$. O limiar δ para a Versão Modificada do Algoritmo Contexto proposta em Galves e Leonardi (2008) também foi analisado e préfixado. Para fins de notação e simplicidade o estimador BIC, o Algoritmo Contexto e a versão alternativa do Algoritmo Contexto proposta em Galves e Leonardi (2008) serão denominados como E_{BIC}, E_{Contex} e E_{Galves}, respectivamente.

2.1 Desempenho de Estimadores de Árvores de Contexto

Nesta seção, verificamos o comportamento das constantes de poda para os estimadores de árvores de contextos, utilizando simulações de amostras não contaminadas. A seguir consideramos duas Árvores Probabilisticas de Contexto cujos contextos e estrutura probabilística foram ilustrados em suas respectivas figuras e tabelas.

2.1.1 Resultados para Árvore 1

Considere a árvore probabilística de contextos apresentada na Figura 4, cujos contextos e estrutura probabilística estão descritos na Tabela 2. Foram simuladas 100 para diferentes tamanhos de amostras, considerando que $\boldsymbol{n}$ representa o tamanho das amostras temos, $n = 500$, $n = 1000$, $n = 5000$, $n = 10000$, $n = 50000$ e $n = 100000$. Para E_{Contex}, a constante C varia dentro do intervalo $\{1.00, 1.01, 1.02, ..., 2.00\}$, para E_{BIC}, a constante C encontra-se no intervalo $\{0.00, 0.01, 0.02, ..., 1.00\}$.

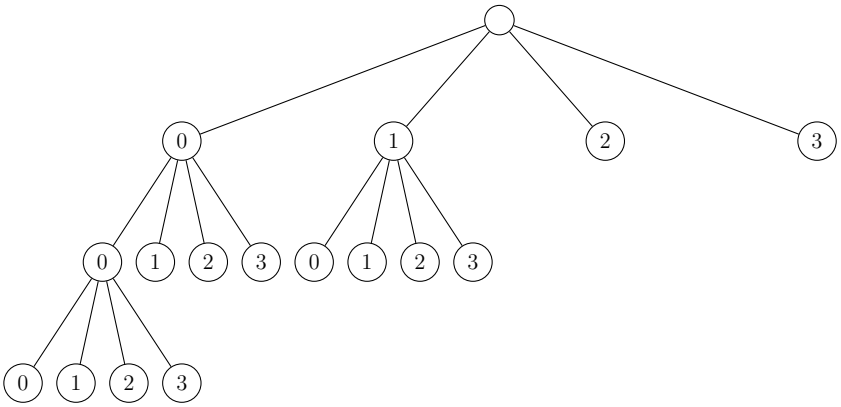


Figura 3 – Árvore 1

Tabela 2 – Probabilidades de Transição da Árvore 1

Contextos	p(0|s)	p(1|s)	p(2|s)	p(3|s)
2	0.3	0.2	0.4	0.1
3	0.35	0.3	0.25	0.1
10	0.4	0.2	0.25	0.15
20	0.35	0.3	0.3	0.05
30	0.05	0.15	0.5	0.3
01	0.15	0.35	0.25	0.25
11	0.3	0.3	0.35	0.05
21	0.05	0.6	0.2	0.15
31	0.05	0.25	0.45	0.25
000	0.2	0.25	0.3	0.25
100	0.0	0.3	0.35	0.35
200	0.2	0.2	0.45	0.15
300	0.0	0.25	0.3	0.45

Figura 4 – Proporção de Acertos do E_{BIC} para Árvore 1

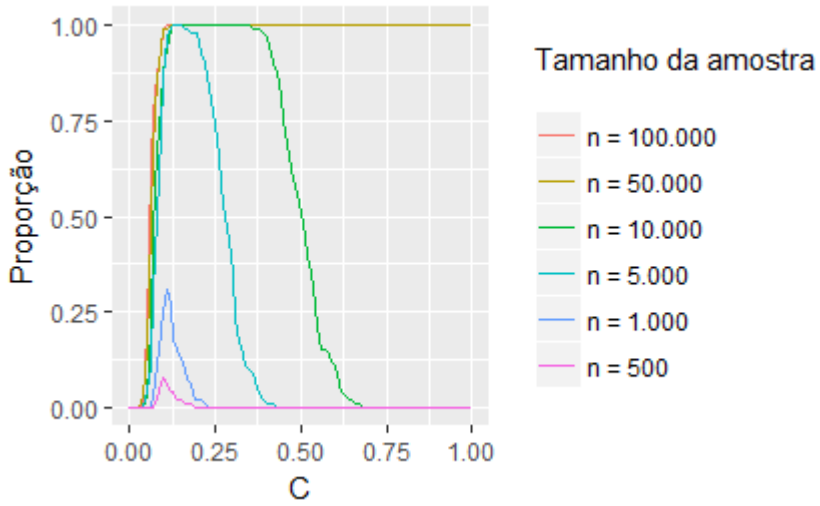
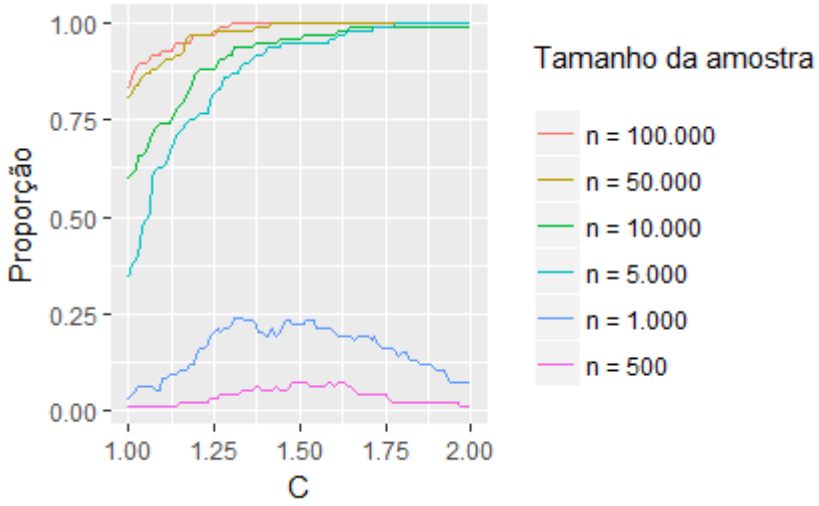


Figura 5 – Proporção de Acertos do E_{Contex} para Árvore 1



2.1.2 Resultados para Árvore 2

Considere agora a árvore probabilística de contextos apresentada na Figura 7, cujos contextos e estrutura probabilística estão descritos na Tabela 3. Foram simuladas 100 para diferentes tamanhos de amostras, considerando que $\boldsymbol{n}$ representa o tamanho das amostras temos, $n = 500$, $n = 1000$, $n = 5000$, $n = 10000$, $n = 50000$ e $n = 100000$. Para E_{Contex}, a constante C varia dentro do intervalo $\{1.00, 1.01, 1.02, ..., 2.00\}$, para E_{BIC}, a constante C encontra-se no intervalo $\{0.00, 0.01, 0.02, ..., 1.00\}$ e para E_{Galves}, a constante δ oscila no intervalo $\{0.00, 0.01, 0.02, ..., 1.00\}$.

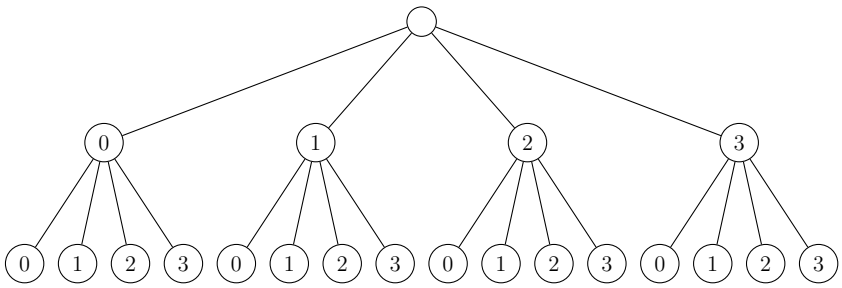


Figura 6 – Árvore 2

Tabela 3 – Probabilidades de Transição da Árvore 2

Contextos	p(0|s)	p(1|s)	p(2|s)	p(3|s)
00	0.3	0.2	0.4	0.1
10	0.4	0.2	0.25	0.15
20	0.35	0.3	0.3	0.05
30	0.05	0.15	0.5	0.3
01	0.15	0.35	0.25	0.25
11	0.3	0.3	0.35	0.05
21	0.05	0.6	0.2	0.15
31	0.05	0.25	0.45	0.25
02	0.2	0.25	0.3	0.25
12	0.0	0.3	0.35	0.35
22	0.2	0.2	0.45	0.15
32	0.0	0.25	0.3	0.45
03	0.0	0.2	0.25	0.55
13	0.2	0.4	0.3	0.1
23	0.25	0.4	0.1	0.25
33	0.05	0.35	0.15	0.45

Figura 7 – Proporção de Acertos do E_{BIC} para Árvore 2

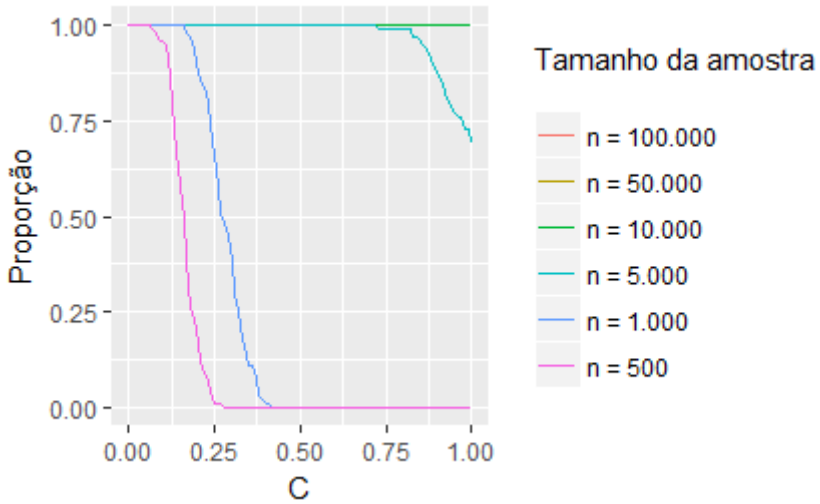


Figura 8 – Proporção de Acertos do E_{Contex} para Árvore 2

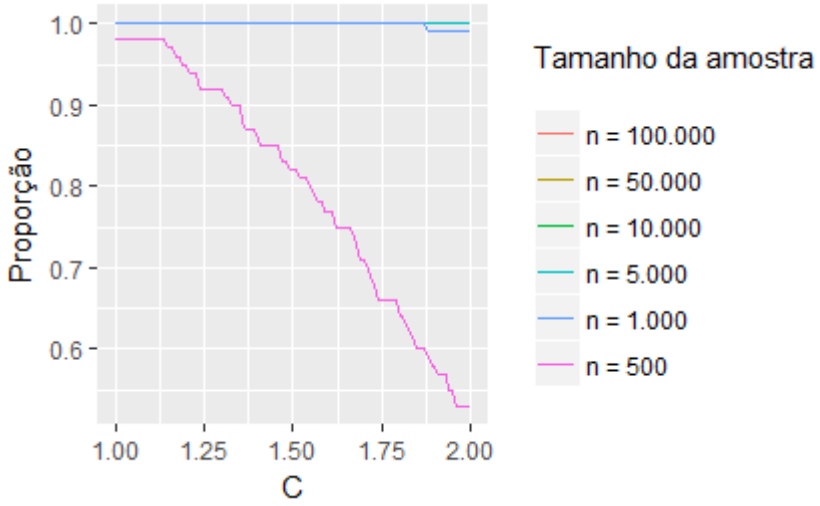
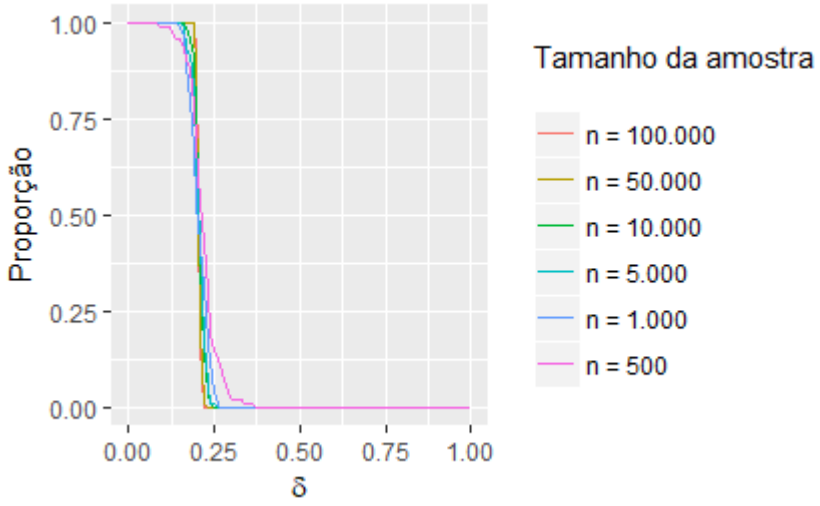


Figura 9 – Proporção de Acertos do E_{Galves} para Árvore 2



2.1.3 Discussão

Vamos primeiramente analisar a seção dos Resultados para Árvore 1.

Considerando o E_{BIC}, para amostra de tamanho $n = 500$, a proporção de acertos máxima foi de apenas 0.08 com constante $C = 0.1$ e 0.31 com constante $C = 0.11$ com amostra de tamanho $n = 1000$. Esses dois valores máximos não foram muito significativos, porém, quando a amostra aumenta, podemos ver outra situação desse estimador. Aplicando em uma amostra com $n = 5000$ esse estimador consegue retornar exatamente a árvore desejada, ou seja, com proporção de identificações corretas atinge seu máximo, que é igual a 1.00 com constante variando entre $\{0.13, 0.14, 0.15\}$. Para $n = 10000$ essa mesma proporção de acertos igual a 1.00 varia sua constante de poda entre $\{0.13, ..., 0.35\}$. Vale ainda ressaltar que, para amostras de tamanho cinco mil ou mais o estimador retornou a proporção de acertos igual a 1.00 a partir da constante de poda igual a 0.1.

Podemos dar uma explicação dos resultados acima analisando a Definição 1.19, o estimador BIC prioriza modelos menos complexos, quanto menor for o valor de C, menor será o valor da constate de penalização c e, consequentem ente, menor será o valor do termo $c|T|\log(n)$. Desse modo, teremos valores menores também para $BIC_{\mathcal{T}}(X_1^n)$. Podemos destacar também que E_{BIC} só apresenta proporções de acertos mais satisfatórios quando o tamanho da amostra é consideravelmente grande. Isso se dá pelo próprio caráter do critério BIC e pela quantidade de parâmetros a serem estimados, ou seja, como árvores mais profundas tendem a possuir mais parâmetros de estimação, isso acarreta um aumento na complexidade do modelo, assim, afetando o estimador. Portanto, o estimador BIC necessita de uma quantidade maior de informação, ou seja, um tamanho de amostra elevado, para recuperar corretamente a árvore desejada.

Já o E_{Contex}, já obtivemos alguns resultados diferentes. Observamos que, para os casos em que a amostra é de tamanho $n = 500$ até $n = 1000$ o estimador não acerta mais que 30% da árvore desejada. Porém com valores de amostra mais elevados ele se torna um bom estimador. Ele é capaz de recuperrar corretamente a árvore desejada, porém necessita de valores de constante de poda maiores, bem como valores maiores de amostras. O estimador consegue retornar a proporção de acertos igual a 1.00 para $n = 5000$ com valores da constante C maiores que 1.78, lembrando que nesse caso $C = \{1.00, 1.01, ..., 2.00\}$. E para amostras maiores que $n = 10000$, $n = 50000$ e $n = 100000$, para que a proporção de acertos seja igual a 1.00 seus valores de C devem ser maiores que, 1.7, 1.42 e 1.3 respectivamente.

Essas características podem ser explicadas pelo fato de que o algoritmo sugerido por Rissanen costuma ser mais permissivo, gerando árvores com maior número de contextos, então para a poda ser mais eficiente, o limiar δ_n precisa ser maior, assim como o tamanho

da amostra. Isso reduziria a quantidade total de contextos e o número de parâmetros a serem estimados.

O estimador E_{Galves} não foi utilizado na seção dos Resultados dos Estimadores para Árvore 1, pois a árvore utilizada fere a condição de não-nulidade, já que, por exemplo, $p(0|100) = 0.0$. Assim, E_{Galves} não retornaria a árvore corretamente, para qualquer tamanho amostral.

De modo geral, os dois estimadores foram muito bons, porém o estimador que obteve melhor desempenho, independentemente do tamanho amostral, foi E_{BIC}, mesmo que o estimador E_{Contex} sendo bem eficiente para retornar a árvore desejada.

Agora passaremos a análise para seção dos Resultados para Árvore 2.

Verificamos um bom desempenho para todos os estimadores. Para E_{Contex} notamos uma convergência muito rápida para encontrar a árvore desejada nessa seção. Todas as árvores foram identificadas corretamente para valores de amostra $n = 1000$ em diante. O E_{Galves} obteve a árvore desejada para constantes de poda dentro do intervalo $\{0.0, ..., 0.19\}$, porém fora desse intervalo não obteve bons resultados. Já o E_{BIC} também recuperou todas as árvores de contexto corretamente. Para $n = 500$ o estimador identificou a árvore desejada dentro do intervalo $C = \{0.0, ..., 0.07\}$, já para valores de $n = 5000$ em diante o estimador tem proporção de acertos igual a 1.00.

Para as árvores completas é interessante observar a forma dos gráficos. Diferentemente dos gráficos para árvores incompletas, observamos que, com constantes de poda com valores pequenos, os estimadores conseguem acertar muito bem a árvore desejada. Por exemplo, podemos ver no gráfico do estimador E_{Galves}, ele varia sua constante de poda com valores entre 0 e 1, porém, no intervalo entre 0 até 0.2 o estimador acerta com proporção de 100% a árvore desejada. Isso ocorre pelo fato da árvore desejada ser uma árvore completa, ou seja, não gostaríamos que a constante de poda podasse muitos galhos da árvore, então quanto menor seu valor, mais acertos os estimadores irão conseguir.

Após a análise dos resultados simulados e sua discussão notamos que a profundidade da árvore, a quantidade de contextos e tamanho do alfabeto afetam diretamente o desempenho dos estimadores tratados no presente trabalho. Para os Resultados para Árvore 1 nos indica que o estimador BIC seria o mais adequado para testar árvores que sejam incompletas e de maior profundidade. Já na seção dos Resultados para Árvore 2 indica que todos os estimadores funcionam com precisão para árvores completas e de profundidade reduzida.

3 Modelagem de Sequências de DNA

Neste capítulo, realizamos estudos sobre doenças cromossomicas e genéticas com objetivo de modelar suas cadeias para análisa-las. Apenas para auxiliar na compreensão, vale ressaltar que as doenças cromossomicas e genéticas neste trabalho são doenças que afetam diretamente um componente genético ou regiões específicas de um cromossomo. Algumas dessas doenças originam-se a partir de uma mutação em um gene, outras podem decorrer de herança. Para fins de simplicidade e nomeclatura, denotaremos essas doenças como doenças cromossomicas.

Selecionamos arquivos que contém sequências de DNA que possuem informações relacionadas as doenças genéticas e cromossomicas. Esses arquivos serão os dados utilizados para aplicação neste trabalho. Esses dados foram obtidos no instituto NCBI (Nacional Center for Biotechnology Information) que conduz pesquisas na área de biologia computacional e estão disponíveis em `https://www.ncbi.nlm.nih.gov/gene/`.

Apenas para melhor compreensão, apresentamos uma breve explicação sobre o que são cromossomos e como são as sequências de DNA. Cromossomos são responsáveis pela transmissão dos caracteres hereditários, ou seja, dos caracteres que são transmitidos de pais para filhos. Os tipos de cromossomos, assim como o número deles, variam de uma espécie para a outra. Nos seres humanos, de maneira geral, apresentam células que possuem 46 cromossomos ou 23 pares. Neste trabalho iremos utilizar um banco de dados que consiste de arquivos com as bases nitrogenadas que formam a sequência de DNA do genoma humano. Sendo que dentro de cada arquivo existe a subdivisão em genes que é específica de uma respectiva doença. O filamento de DNA de cada um dos genes é formado por duas fitas (sequências) de bases nitrogenadas ligadas umas as outras. As bases são apresentadas abaixo juntamente com suas respectivas letras iniciais:

- Adenina = A.
- Citosina = C.
- Guanina = G.
- Timina = T.

O arquivo do banco de dados contém apenas um lado da fita de DNA, já que o outro lado pode ser obtido observando as bases com suas respectivas ligações, que são:

- A se liga com T e vice versa (A – T e T – A);

- C se liga com G e vice versa (C – G e G – C).

Selecionamos 30 arquivos e os codificamos da maneira abaixo, sendo que só consideramos arquivos que, após a codificação, resultaram em sequências de tamanho maior que dez mil. Cada arquivo foi codificado em uma sequência de inteiros tomando valores no alfabeto $\mathcal{A} = \{0, 1, 2, 3\}$ da seguinte forma:

- Adenina (A) = 0;
- Timina (T) = 1;
- Guanina (G) = 2;
- Citosina (C) = 3.

Ao pesquisar os arquivos para nosso banco de dados notamos que muitas doenças estão localizadas nos mesmos cromossomos, ou nos mesmos braços genéticos, ou ainda compartilhando parte do mesmo código. Assim podemos ver que existem caracteristicas semelhantes entre algumas delas. Queremos então modelar uma árvore que seja mais identificada entre os arquivos selecionados.

Logo, o alfabeto que será utilizado neste trabalho será $\mathcal{A} = \{0, 1, 2, 3\}$. Podemos demonstrar um exemplo da estimação de uma árvore de um cromossomo com ordem 3 e número de contextos igual a 10. A árvore da esquerda representa os dados não codificados e a direita temos a árvore codificada.

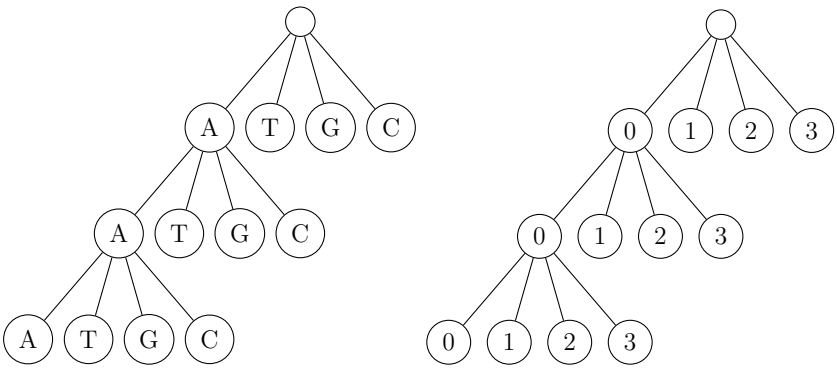


Figura 10 – Representações de árvores de contextos

Baseado-se no capítulo anterior dos Estudos Computacionais, aplicamos o estimador BIC, que foi o mais eficiente entre os estimadores, em todos as doenças cromossomicas e observamos suas probabilidades de transição, bem como as árvores que mais são identificadas. Como a média do tamanho das amostras é maior que $n = 50000$ verificamos que uma constante de poda com um valor maior apresentava um melhor ajuste para encontrarmos as árvores mais comuns. Após alguns testes utilizamos a constante de poda $C = 0.65$

e com profundidade $d = 3$ para aplicarmos o estimador. O algoritmo usado retornou 5 árvores diferentes, dispostas abaixo com a notação Árvore A, Árvore B, Árvore C, Árvore D e Árvore E. A frequência para o ocorrimento de cada árvore está contida na Tabela 4.

Tabela 4 – Frequência das árvores encontradas pelo BIC com $C = 0.65$ e $d = 3$

Árvores	Frequência
Árvore A	7
Árvore B	19
Árvore C	1
Árvore D	2
Árvore E	1

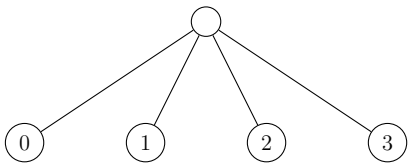


Figura 11 – Árvore A

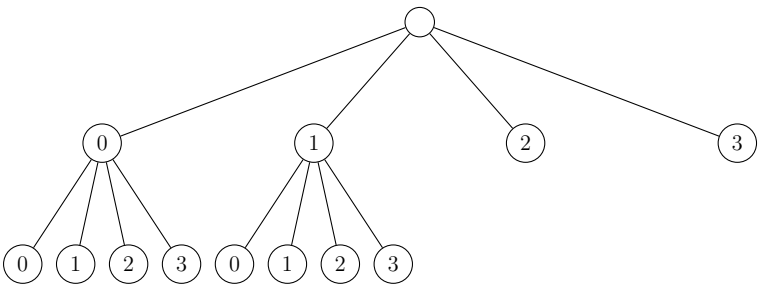


Figura 12 – Árvore B

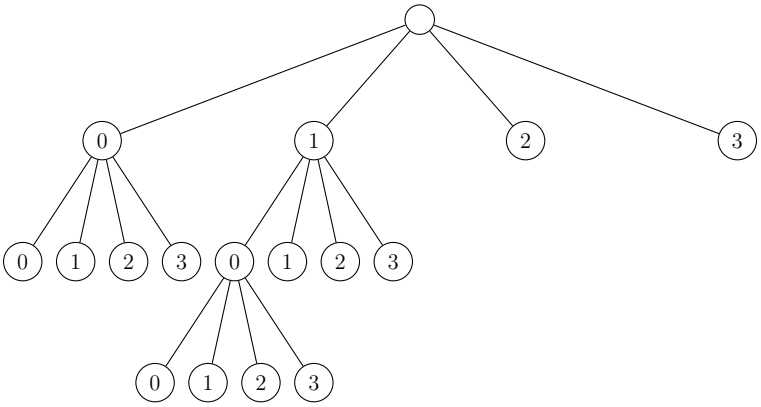


Figura 13 – Árvore C

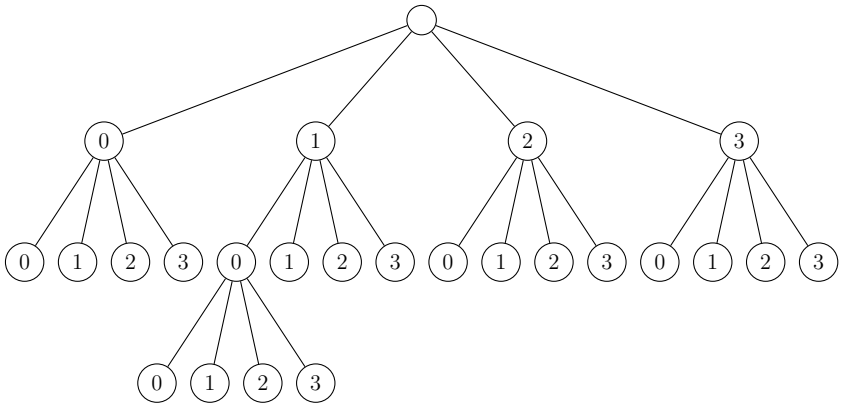


Figura 14 – Árvore D

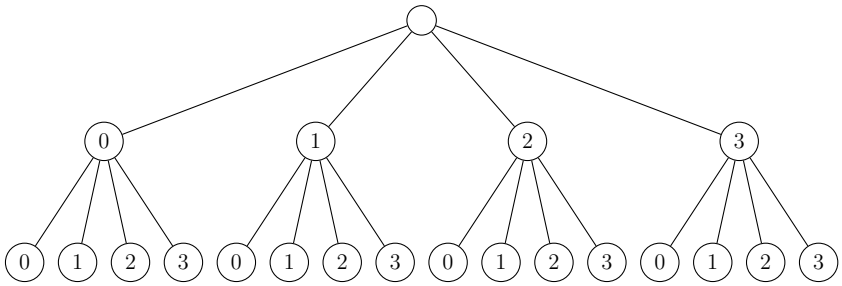


Figura 15 – Árvore E

Como cada doença cromossomica, que representa cada arquivo utilizado nesse trabalho, gera sua própria probabilidade de transição, supomos que seria inviável apresentar trinta tabelas. Então pegamos um arquivo aleatório para servir apenas como exemplo. Fizemos as probabilidades de transição para um dos arquivos de acordo com os Contextos da Árvore B, que foi a que se repetiu com maior frequência.

Tabela 5 – Probabilidades de Transição do BIC para $C = 0.65$ da Árvore B

Contextos	0	1	2	3
00	0,367	0,245	0,226	0,162
10	0,316	0,293	0,213	0,178
20	0,311	0,218	0,310	0,161
2	0,263	0,255	0,252	0,229
30	0,213	0,240	0,330	0,217
3	0,316	0,336	0,071	0,278
01	0,202	0,327	0,266	0,205
11	0,180	0,393	0,219	0,209
21	0,185	0,272	0,335	0,208
31	0,153	0,285	0,326	0,236

Sobre os contextos e as árvores geradas pelo estimador usado, podemos fazer algumas considerações. As Árvores A e B obtiveram as maiores frequências de identificação. A

Árvore A apareceu em 23% dos casos, já a Árvore B foi a mais comum, sendo identificada em 63.3% das doenças cromossomicas. Os contextos $00, 10, 20, 30, 11, 21, 31$ apareceram em todos os arquivos analisados, exceto na Árvore A, que apresenta profundidade igual a 1. Não houve um contexto que se repetiu para todos os casos.

Outro teste do estimador BIC nos arquivos que repesentam as doenças cromossomicas foi aplicado. Agora variamos a constante d, que representa a profundidade da árvore. Variamos para valores maiores que $d = 3$, como $d = 4$, $d = 5$ e $d = 6$, mas os resultados foram os mesmos. Porém quando subtituimos a profundidade por $d = 2$ e mantivemos a constante de poda $C = 0.65$, obtivemos os seguintes resultados apresentados da Tabela 4.

Tabela 6 – Frequência das árvores encontradas pelo BIC com $C = 0.65$ e $d = 2$

Árvores	Frequência
Árvore A	7
Árvore B	20
Árvore E	3

Realizadas as modelagens, podemos ter algumas percepções. Podemos ver que a árvore com maior proporção de identificações, Árvore B, possui ramificações nos galhos 0 e 1, ou seja, quando observamos a caracteristica dessa árvore nos dados de doenças cromossomicas, podemos ver que, não é necessário olhar outros passos para trás quando temos as informações 2 e 3, isto é, Guanina e Citosina, respectivamente. Podemos calcular as probabilidades de transição do próximo símbolo apenas olhando um passo atrás. Já quando analisamos os galhos 0 e 1, Adenina e Timina, respectivamente, podemos ver que é necessário olhar mais um passo para trás, ou seja, observando o símbolo anterior trazendo uma profundidade $d = 2$ para a árvore.

Uma característica dos modelos de ordem variável é que em muitas aplicações a forma da árvore de contextos tem uma interpretação natural e informativa. Pela árvore de contextos estimada e apresentada na Figura 12, podemos predizer se o próximo símbolo, apenas considerando no máximo as informações de dois passos atrás. Outra característica dos modelos é a representação gráfica do processo em forma de uma árvore, nos dá a informação de como o processo se comporta, ou se sofreu alterações no decorrer dos passos.

Vale também analisar que é interessante ver como essas doenças carregam características em comum, e nesse caso, uma das características foi apresentada pelos contextos mais identificados nesse estudo. Foge da intenção deste trabalho analisar o que esse conjunto de bases nitrogenadas, representadas por esses contextos, significam e como afetam em suas respectivas doenças, mas podem vir a ser informações importantes para campos da área da saúde e genética.

4 Considerações Finais

Neste trabalho, estudamos as Cadeias de Ordem Variável, aplicamos e avaliamos os estimadores do Algoritmo Contexto, da Versão Modificada do Algoritmo Contexto e do BIC e implementamos uma aplicação modelando sequências de DNA relacionadas a doenças cromossomicas. Após esses procedimentos concluídos, compreendemos que as Cadeias de Ordem Variável são uma classe de modelos bastante inovadora, dinâmica, adaptável e competente em suas aplicações que podem ser em diversas áreas.

No Capítulo 1, apresentamos as Definições e Notações Básicas que trouxe uma base teórica para entendimento deste trabalho. Porém, com o decorrer dos estudos, notamos que existe uma quantidade enorme de conceitos e fundamentos a serem estudados e que podem complementar este e vários outros trabalhos com mesmo tema. Entretanto, seria inviável abordar tantos conceitos e poderiamos desviar do foco deste trabalho.

No Capítulo 2, onde apresentamos os Estudos Computacionais, analisamos o desempenho dos estimadores e de suas respectivas constantes abordadas neste trabalho, o que pode, mesmo não sendo algo muito significativo, trazer algum avanço para o desenvolvimento dos temas tratados. Mesmo assim, ressaltamos novamente a eficiência do estimador BIC que mostrou ser o mais apropriado para análise de árvores de contexto, especialmente para árvores incompletas e mais profundas em casos com tamanhos de amostras grandes.

No Capítulo 3, dos Estudos Aplicados, trouxemos uma breve explicação genética e a estrutura de como seria realizado os estudos neste trabalho. Consideramos um alfabeto $\mathcal{A} = \{0, 1, 2, 3\}$, realizamos uma modelagem das sequências de DNA relacionadas a doenças cromossomicas. Observamos que existem infinitas formas de aplicações, estudos e análises nesse tema. Uma ideia interessante seria encontrar um modelo que representasse diferentes doenças presentes em um único cromossomo, análisar suas características entre elas e comparar o modelo encontrado com mais frequência entre as doenças com um modelo que representaria todo o cromossomo estudado. Essa análise poderia trazer informações importantes para encontrar características específicas entre o cromossomo e as doenças analisadas. Esse estudo fica como sugestão a trabalhos futuros.

5 Referências Bibliográficas

[1] Bomfim, A., Estudo de Estimadores de Árvores de Contexto com Aplicação em Linguística. Monografia apresentada para obtenção do título de Bacharel em Estatística. Brasília, 2015.

[2] Bühlmann, P. and Wyner, A. J., Variable length Markov chains, Ann. Statist. 27, 480-513, 1999.

[3] Csiszár, I. e Talata, Z., Context tree estimation for not necessarily finite memory processes, via BIC and MDL, IEEE Trans. Inform. Theory 52, Number3, 1007-1016, 2006.

[4] Duarte, D., Modelagem Estocástica de Sequências de DNA Através de Cadeias de Memória de Alcance Variável, 2010.

[5] Ferrari F. e Wyner A., Estimation of general stationary processes by variable length Markov chains, Scand. J. Statist. 30, no. 3, 459-480, 2003.

[6] Galves, A., Galves, C., Garcia, J. E., Garcia, N. L. and Leonardi, F., Context tree selection and linguistic rhythm retrieval from written texts, Annals of Applied Statistics, 6 1, 186-209, 2012.

[7] Galves, A. Leonardi, F., Exponential inequalities for empirical unbounded context trees. Progress in Probability 60, 257-270, 2008.

[8] Garcia, N. L., Moreira L., Stochastically Perturbed Chains of Variable Memory, Springer, New York, 2015.

[9] Matta, D., Algoritmos de estimação para Cadeias de Markov de Alcance Variável - aplicações a detecção do ritmo em textos escritos. Dissertação de Mestrado (Mestre em Estatística) - Instituto de Matemática, Estatística e Computação Cientifica, Universidade Estadual de Campinas, Campinas, 2008.

[10] Moreira, L. Processos de Ordem Infinita Estocasticamente Perturbados. Tese (Doutorado em Estatística) - Instituto de Matemática,Estatística e Computação Cientifica, Universidade Estadual de Campinas, Campinas, 2012.

[11] Quintino, F. Aplicações de Cadeias de Ordem Variável Estocasticamente Perturbadas. Monografia apresentada para obtenção do título de Bacharel em Estatística. Brasília, 2015.

[12] Rissanen, J., A universal data compression system, IEEE Trans. Inform. Theory 29(5): 656-664, 1983.

Apêndices

Esse apêndice contem os códigos das funções utilizadas neste trabalho que foram implementadas no software R para o Algoritmo Contexto, a Versão Modificada do Algoritmo Contexto proposta por Galves e Leonardi (2008) e o estimador BIC.

Apêndice A - Códigos para os algoritmos

```
############### FUNÇÃO BIC ##############################

fbic <- function(dados,d,constante,perturbacao = 0){

  # perturbação

  if (perturbacao != 0){
    for (i in 1:length(dados)){
      dados[i] <- dados[i]*rbinom(1,1,1-perturbacao)
    }
  }

  # valor de |A| (considerando A = {0,1,...,|A| - 1})

  alfabeto <- max(dados)+1

  # função para converter (i,j) em sequência

  fconverte <- function(i,j){
    conversao <- character(1)
    for (l in (d+1-j):1){
      conversao <- paste(floor((i-1)/alfabeto^(l-1)),conversao,sep="")
      i <- ((i-1) %% alfabeto^(l-1)) + 1
    }
    conversao
  }

  # função para completar matriz
```

```
fcompleta <- function(matriz,q) {
  d <- (ncol(matriz)-1)
  for (j in 1:d){
    for (i in 1:(q^d)){
      matriz[floor((i-1)/q)+1,j+1] <- matriz[floor((i-1)/q)+1,j+1] + matriz[i,j]
    }
  }
  matriz
}

# contagem de N_n(s,a) e N_n(s)

num <- array(0,c(alfabeto^d,d+1,alfabeto))

for (tempo in (d+1):length(dados)){
  i <- 0
  ajuste <- 0
  for (passado in (tempo-d):(tempo-1)){
    i <- i + (alfabeto^ajuste)*(dados[passado])
    ajuste <- ajuste + 1
  }
  num[i+1,1,(dados[tempo])+1] <- num[i+1,1,(dados[tempo])+1] + 1
}

for (i in 1:alfabeto){
  num[,,i] <- fcompleta(num[,,i],alfabeto)
}

numt <- apply(num,c(1,2),sum)

# probabilidades de transição estimadas

tr <- array(0,c(alfabeto^d,d+1,alfabeto))

for (j in 1:(d+1)){
  for (i in 1:alfabeto^(d+1-j)){
    for (k in 1:alfabeto){
      if (numt[i,j] == 0) tr[i,j,k] <- 1/alfabeto
      else tr[i,j,k] <- num[i,j,k]/numt[i,j]
```

```
    }
  }
}

# achando a matriz v e a matriz x

matriz <- matrix(0,alfabeto^d,d+1)
matrizv <- matrix(0,alfabeto^d,d+1)
matrizx <- matrix(0,alfabeto^d,d+1)

for (j in 1:(d+1)){
  for (i in 1:alfabeto^(d+1-j)){
    for (k in 1:alfabeto){
      if (tr[i,j,k] != 0){
        matriz[i,j] <- matriz[i,j] + num[i,j,k]*log(tr[i,j,k])
      }
      matriz[i,j] <- matriz[i,j] - constante*log(length(dados))
      if (numt[i,j] == 0) matriz[i,j] <- 0
    }
  }
}

for (i in 1:alfabeto^d){
  matrizv[i,1] <- matriz[i,1]
}
for (j in 2:(d+1)){
  for (i in 1:alfabeto^(d+1-j)){
    for (anterior in 1:alfabeto){
      matrizv[i,j] <- matrizv[i,j] + matriz[(i-1)*alfabeto+anterior,j-1]
    }
    if (matriz[i,j] > matrizv[i,j]){
      matrizv[i,j] <- matriz[i,j]
    }
  }
}

for (j in 1:(d+1)){
  for (i in 1:alfabeto^(d+1-j)){
    matrizx[i,j] <- as.integer(matrizv[i,j] > matriz[i,j])
```

```
    }
  }

  # achando a árvore

  arvore <- character()
  arvore[1] <- "sequencia vazia"
  index <- 1
  valor <- 0
  for (i in 1:alfabeto^d){
    for (j in 1:d){
      valor <- 0
      sufixo <- i
      while (matrizx[i,j] == 0 && matrizx[floor((sufixo-1)/alfabeto)+1,j+valor+1] == 1){
        valor <- valor + 1
        sufixo <- floor((sufixo-1)/alfabeto+1)
        if (j+valor > d) break
      }
      if (valor == d-j+1 && numt[i,j] > 0){
        arvore[index] <- fconverte(i,j)
        index <- index + 1
      }
    }
  }
  arvore
}


######################   FUNÇÃO CONTEXTO RISSANEN    ###################################


fcontexto <- function(dados,d,constante,perturbacao = 0){
  # perturbação
  if (perturbacao != 0){
    for (i in 1:length(dados)){
      dados[i] <- dados[i]*rbinom(1,1,1-perturbacao)
    }
  }
```

```
# valor de |A| (considerando A = {0,1,...,|A| - 1})
alfabeto <- max(dados)+1
# limiar delta_n
limiar <- constante*log(length(dados))
# função para converter (i,j) em sequência
fconverte <- function(i,j){
  conversao <- character(1)
  for (l in (d+1-j):1){
    conversao <- paste(floor((i-1)/alfabeto^(l-1)),conversao,sep="")
    i <- ((i-1) %% alfabeto^(l-1)) + 1
  }
  conversao
}

fcompleta <- function(matriz,q) {
  d <- (ncol(matriz)-1)
  for (j in 1:d){
    for (i in 1:(q^d)){
      matriz[floor((i-1)/q)+1,j+1] <- matriz[floor((i-1)/q)+1,j+1] + matriz[i,j]
    }
  }
  matriz
}
# contagem de N_n(s,a) e N_n(s)
num <- array(0,c(alfabeto^d,d+1,alfabeto))
for (tempo in (d+1):length(dados)){
  i <- 0
  ajuste <- 0
  for (passado in (tempo-d):(tempo-1)){
    i <- i + (alfabeto^ajuste)*(dados[passado])
    ajuste <- ajuste + 1
  }
  num[i+1,1,(dados[tempo])+1] <- num[i+1,1,(dados[tempo])+1] + 1
}
for (i in 1:alfabeto){
  num[,,i] <- fcompleta(num[,,i],alfabeto)
}
numt <- apply(num,c(1,2),sum)
# probabilidades de transição estimadas
```

```
tr <- array(0,c(alfabeto^d,d+1,alfabeto))
for (j in 1:(d+1)){
  for (i in 1:alfabeto^(d+1-j)){
    for (k in 1:alfabeto){
      if (numt[i,j] == 0) tr[i,j,k] <- 1/alfabeto
      else tr[i,j,k] <- num[i,j,k]/numt[i,j]
    }
  }
}
# função divergência
fdiv <- function(i,j,b){
  p <- rep(0,alfabeto)
  q <- rep(0,alfabeto)
  for (a in 1:alfabeto){
    p[a] <- p[a] + tr[(i-1)*alfabeto+b,j-1,a]
    q[a] <- q[a] + tr[i,j,a]
  }
  div <- numeric(alfabeto)
  for (a in 1:alfabeto){
    if (p[a] == 0) div[a] <- 0
    else if (q[a] == 0) div[a] <- Inf
    else div[a] <- p[a]*log(p[a]/q[a])
  }
  sum(div)
}
# função Delta
fdelta <- function(i,j){
  delta <- 0
  for (b in 1:alfabeto){
    if (numt[(i-1)*alfabeto+b,j-1] != 0){
      delta <- delta + numt[(i-1)*alfabeto+b,j-1]*fdiv(i,j,b)
    }
  }
  delta
}
# achando a matriz C
matrizc <- matrix(0,alfabeto^d,d+1)
for (l in (d-1):0){
  for (i in 1:(alfabeto^l)){
```

```
      for (proximo in 1:alfabeto){
        if (matrizc[(i-1)*alfabeto+proximo,(d-l)] == 1){
          matrizc[i,(d+1-l)] <- 1
        }
      }
      if (matrizc[i,(d+1-l)] == 0 && numt[i,(d+1-l)] >= 1){
        matrizc[i,(d+1-l)] <- as.integer(fdelta(i,d-l+1) > limiar)
      }
    }
  }
  # achando a árvore
  arvore <- character()
  arvore[1] <- "sequencia vazia"
  index <- 1
  valor <- 0
  for (i in 1:alfabeto^d){
    for (j in 1:d){
      valor <- 0
      if (matrizc[i,j] == 0 && matrizc[floor((i-1)/alfabeto)+1,j+1] == 1){
        valor <- 1
      }
      if (valor == 1 && numt[i,j] != 0){
        arvore[index] <- fconverte(i,j)
        index <- index + 1
      }
    }
  }
  resultado<-as.list(0)
  resultado[[1]]<-arvore
  resultado[[2]]<-num
  resultado
}

################## CONTEXTO MODIFICADA ################################

fgalves <- function(dados,d,delta,perturbacao = 0){
  # pertuba¸c~ao
  if (perturbacao != 0){
    for (i in 1:length(dados)){
```

```
    dados[i] <- dados[i]*rbinom(1,1,1-perturbacao)
  }
}
# valor de |A| (considerando A = {0,1,...,|A| - 1})
alfabeto <- max(dados)+1
# fun¸c~ao para converter (i,j) em sequ^encia
fconverte <- function(i,j){
  conversao <- character(1)
  for (l in (d+1-j):1){
    conversao <- paste(floor((i-1)/alfabeto^(l-1)),conversao,sep="")
    i <- ((i-1) %% alfabeto^(l-1)) + 1
  }
  conversao
}
# fun¸c~ao para completar matriz
fcompleta <- function(matriz,q) {
  d <- (ncol(matriz)-1)
  45
  for (j in 1:d){
    for (i in 1:(q^d)){
      matriz[floor((i-1)/q)+1,j+1] <- matriz[floor((i-1)/q)+1,j+1] + matriz[i,j]
    }
  }
  matriz
}
# contagem de N_n(s,a) e N_n(s)
num <- array(0,c(alfabeto^d,d+1,alfabeto))
for (tempo in (d+1):length(dados)){
  i <- 0
  ajuste <- 0
  for (passado in (tempo-d):(tempo-1)){
    i <- i + (alfabeto^ajuste)*(dados[passado])
    ajuste <- ajuste + 1
  }
  num[i+1,1,(dados[tempo])+1] <- num[i+1,1,(dados[tempo])+1] + 1
}
for (i in 1:alfabeto){
  num[,,i] <- fcompleta(num[,,i],alfabeto)
}
```

```
numt <- apply(num,c(1,2),sum)
# probabilidades de transi¸c~ao estimadas

tr <- array(0,c(alfabeto^d,d+1,alfabeto))
for (j in 1:(d+1)){
  for (i in 1:alfabeto^(d+1-j)){
    for (k in 1:alfabeto){
      if (numt[i,j] == 0) tr[i,j,k] <- 1/alfabeto
      else tr[i,j,k] <- num[i,j,k]/numt[i,j]
    }
  }
}
# matriz com os Deltas
adelta <- array(0,c(alfabeto^d,d,alfabeto))
for(a in 1:alfabeto){
  for (j in 1:d){
    for (i in 1:alfabeto^(d-j+1)){
      adelta[i,j,a] <- abs(tr[i,j,a]-tr[floor((i-1)/alfabeto)+1,j+1,a])
    }
  }
}
mdelta <- apply(adelta,c(1,2),max)
# achando a matriz de zeros e uns
matriz <- matrix(0,alfabeto^d,d+1)
for (j in 1:d){

  for (i in 1:alfabeto^(d-j+1)){

    if (matriz[i,j] == 1){
      matriz[floor((i-1)/alfabeto)+1,j+1] <- 1
    }
    else if (matriz[floor((i-1)/alfabeto)+1,j+1] == 0){
      matriz[floor((i-1)/alfabeto)+1,j+1] <- as.integer(mdelta[i,j] > delta)
    }
  }
}
# achando a arvore
arvore <- character()
arvore[1] <- "sequencia vazia"
```

```
  index <- 1
  valor <- 0
  for (i in 1:alfabeto^d){
    for (j in 1:d){
      valor <- 0
      if (matriz[i,j] == 0 && matriz[floor((i-1)/alfabeto)+1,j+1] == 1){
        valor <- 1
      }
      if (valor == 1){
        arvore[index] <- fconverte(i,j)
        index <- index + 1
      }
    }
  }
  arvore
}
```

Apêndice B - Códigos para as aplicações nos dados

```
setwd("C:/Users/ghcm1/Desktop/UnB/Semestre 2018 2/TCC 2/Dados/DoençasGeneticas/Genes_Cod
DadosGenet <- list() # creates a list
listfiles <- dir(pattern = ".txt")
for (k in 1:length(listfiles)){
  DadosGenet[[k]] <- as.vector(scan(listfiles[k], what = 'character', quote = ""))
}
for (k in 1:length(DadosGenet)) {
  DadosGenet[[k]] <- strsplit(DadosGenet[[k]], "")
}

for (k in 1:length(DadosGenet)) {
  DadosGenet[[k]][[1]] <- as.numeric(DadosGenet[[k]][[1]])
}

for (k in 1:length(DadosGenet)) {
  DadosGenet[[k]][[1]] <- fbic(DadosGenet[[k]][[1]], d = 3, constante = 0.65)
}

unique(DadosGenet)
```

```
tabulate(match(DadosGenet, unique(DadosGenet)))
```

Apêncice C - Código aplicado nos estudos computacionais

Esse código mostra a ideia da implementação dos estudos aplicados para o estimador BIC com tamanho de amostra igual a quinhentos, para uma árvore incompleta e a criação do gráfico para comparação.

```
tamanho = 500
dados = vector("integer", length = tamanho)
dados[c(1,2,3,4)] = c(0,1,2,3)
dados = rep(dados, times = 100)
dados2 = split(dados, ceiling(seq_along(dados)/tamanho))
sequ = seq(0, 1, 0.01)
lista.bic = rep(list(list()), length(sequ))

P= c("000" ,"100" ,"10", "200" ,"20"  ,"2" ,  "300", "30",  "3", "01", "11", "21",

for (i in 5:tamanho) {
  for(k in 1:100) {
    v = runif(1)

    if(dados2[[k]][[i-1]] == 2){
      if(v < a2) {
        dados2[[k]][[i]] = 0
      } else if(v < b2) {
        dados2[[k]][[i]] = 1
      } else if(v < c2) {
        dados2[[k]][[i]] = 2
      } else {
        dados2[[k]][[i]] = 3
      }
    }
    if(dados2[[k]][[i-1]] == 3){
      if(v < a3) {
        dados2[[k]][[i]] = 0
      } else if(v < b3) {
        dados2[[k]][[i]] = 1
      } else if(v < c3) {
        dados2[[k]][[i]] = 2
```

```
  } else {
    dados2[[k]][[i]] = 3
  }
}
if(dados2[[k]][[i-2]] == 1 && dados2[[k]][[i-1]] == 0){
  if(v < a10) {
    dados2[[k]][[i]] = 0
  } else if(v < b10) {
    dados2[[k]][[i]] = 1
  } else if(v < c10) {
    dados2[[k]][[i]] = 2
  } else {
    dados2[[k]][[i]] = 3
  }
}
if(dados2[[k]][[i-2]] == 2 && dados2[[k]][[i-1]] == 0){
  if(v < a20) {
    dados2[[k]][[i]] = 0
  } else if(v < b20) {
    dados2[[k]][[i]] = 1
  } else if(v < c20) {
    dados2[[k]][[i]] = 2
  } else {
    dados2[[k]][[i]] = 3
  }
}
if(dados2[[k]][[i-2]] == 3 && dados2[[k]][[i-1]] == 0){
  if(v < a30) {
    dados2[[k]][[i]] = 0
  } else if(v < b30) {
    dados2[[k]][[i]] = 1
  } else if(v < c30) {
    dados2[[k]][[i]] = 2
  } else {
    dados2[[k]][[i]] = 3
  }
}
if(dados2[[k]][[i-2]] == 0 && dados2[[k]][[i-1]] == 1){
  if(v < a01) {
```

```
      dados2[[k]][[i]] = 0
    } else if(v < b01) {
      dados2[[k]][[i]] = 1
    } else if(v < c01) {
      dados2[[k]][[i]] = 2
    } else {
      dados2[[k]][[i]] = 3
    }
  }
  if(dados2[[k]][[i-2]] == 1 && dados2[[k]][[i-1]] == 1){
    if(v < a11) {
      dados2[[k]][[i]] = 0
    } else if(v < b11) {
      dados2[[k]][[i]] = 1
    } else if(v < c11) {
      dados2[[k]][[i]] = 2
    } else {
      dados2[[k]][[i]] = 3
    }
  }
  if(dados2[[k]][[i-2]] == 2 && dados2[[k]][[i-1]] == 1){
    if(v < a21) {
      dados2[[k]][[i]] = 0
    } else if(v < b21) {
      dados2[[k]][[i]] = 1
    } else if(v < c21) {
      dados2[[k]][[i]] = 2
    } else {
      dados2[[k]][[i]] = 3
    }
  }
  if(dados2[[k]][[i-2]] == 3 && dados2[[k]][[i-1]] == 1){
    if(v < a31) {
      dados2[[k]][[i]] = 0
    } else if(v < b31) {
      dados2[[k]][[i]] = 1
    } else if(v < c31) {
      dados2[[k]][[i]] = 2
    } else {
```

```
    dados2[[k]][[i]] = 3
  }
}

if(dados2[[k]][[i-3]] == 0 && dados2[[k]][[i-2]] == 0 && dados2[[k]][[i-1]] == 0){
  if(v < a000) {
    dados2[[k]][[i]] = 0
  } else if(v < b000) {
    dados2[[k]][[i]] = 1
  } else if(v < c000) {
    dados2[[k]][[i]] = 2
  } else {
    dados2[[k]][[i]] = 3
  }
}
if(dados2[[k]][[i-3]] == 1 && dados2[[k]][[i-2]] == 0 && dados2[[k]][[i-1]] == 0){
  if(v < a100) {
    dados2[[k]][[i]] = 0
  } else if(v < b100) {
    dados2[[k]][[i]] = 1
  } else if(v < c100) {
    dados2[[k]][[i]] = 2
  }  else {
    dados2[[k]][[i]] = 3
  }
}
if(dados2[[k]][[i-3]] == 2 && dados2[[k]][[i-2]] == 0 && dados2[[k]][[i-1]] == 0){
  if(v < a200) {
    dados2[[k]][[i]] = 0
  } else if(v < b200) {
    dados2[[k]][[i]] = 1
  } else if(v < c200) {
    dados2[[k]][[i]] = 2
  } else {
    dados2[[k]][[i]] = 3
  }
}
if(dados2[[k]][[i-3]] == 3 && dados2[[k]][[i-2]] == 0 && dados2[[k]][[i-1]] == 0){
  if(v < a300) {
```

```
        dados2[[k]][[i]] = 0
      } else if(v < b300) {
        dados2[[k]][[i]] = 1
      } else if(v < c300) {
        dados2[[k]][[i]] = 2
      } else {
        dados2[[k]][[i]] = 3
      }
    }

  }
}

for (j in 1:100) {
  for (i in 1:length(sequ)) {
    lista.bic[[i]][[j]] = fbic(dados2[[j]],d = 3,constante = sequ[i])
  }
}

comparacoes <- rep(list(list()), length(sequ))

for(i in 1:length(sequ)){
  for(j in 1:100)
    comparacoes[[i]][[j]] = identical(lista.bic[[i]][[j]], P)
}

acertos = rep(list(list()), length(sequ))

for(i in 1:length(sequ)){
  acertos[[i]] = unlist(comparacoes[[i]])
}

for(i in 1:length(sequ)){
  acertos[[i]] = sum(acertos[[i]], na.rm = T)
}



bic500 = unlist(acertos)
bic500=bic500/100
```

```
names(bic500) = sequ
cbind(bic500)

require("dplyr")
require("ggplot2")
plotar <- data.frame(delta = sequ, proporcao = bic500)
plotar %>%
  ggplot(aes(x = delta, y = bic500)) +
  geom_point() + geom_line() +
  labs(y = 'Proporção', x = 'C') +
  ggtitle('Proporção de Acertos da BIC N=500')
```