



UNIVERSIDADE FEDERAL DO PARÁ  
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS  
PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA E ESTATÍSTICA

# IMPLEMENTAÇÃO E OTIMIZAÇÃO DO PACOTE *TestFraud* PARA DETECÇÃO DE FRAUDE EM TESTES

Miguel Monteiro de Souza

Orientação: Prof. Dr. Héilton Ribeiro Tavares  
Coorientação: Profa. Dra. Maria Regina Madruga Tavares

*O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001*

Belém  
2019

Miguel Monteiro de Souza

**IMPLEMENTAÇÃO E OTIMIZAÇÃO DO  
PACOTE *TestFraud* PARA DETECÇÃO DE  
FRAUDE EM TESTES**

Dissertação apresentada ao Curso de Mestrado em Matemática e Estatística da Universidade Federal do Pará, como pré-requisito para a obtenção do título de Mestre em Estatística.

Orientação: **Prof. Dr. Héilton Ribeiro Tavares**

Coorientação: **Profa. Dra. Maria Regina Madruga Tavares**

**Belém**

**2019**

**Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD  
Sistema de Bibliotecas da Universidade Federal do Pará  
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)**

---

M772i Monteiro de Souza, Miguel  
Implementação e otimização do pacote TestFraud para detecção  
de fraude em testes / Miguel Monteiro de Souza. — 2019.  
55 f. : il. color.

Orientador(a): Prof. Dr. Héilton Ribeiro Tavares  
Coorientação: Prof<sup>a</sup>. Dra. Maria Regina Madruga Tavares  
Dissertação (Mestrado) - Programa de Pós-Graduação em  
Matemática e Estatística, Instituto de Ciências Exatas e Naturais,  
Universidade Federal do Pará, Belém, 2019.

1. Processamento paralelo. 2. Métodos para detecção de  
fraude em testes. 3. Avaliação em larga escala. 4. Teoria da  
Resposta ao Item. I. Título.

CDD 310

---

Miguel Monteiro de Souza

IMPLEMENTAÇÃO E OTIMIZAÇÃO DO PACOTE *TestFraud* PARA  
DETECÇÃO DE FRAUDE EM TESTES

Esta Dissertação foi julgada e aprovada para a obtenção do grau de Mestre em Estatística, no Programa de Pós-Graduação em Matemática e Estatística da Universidade Federal do Pará.

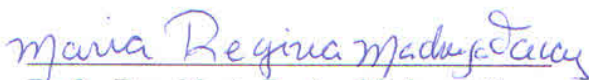
Belém, 20 de Março de 2019

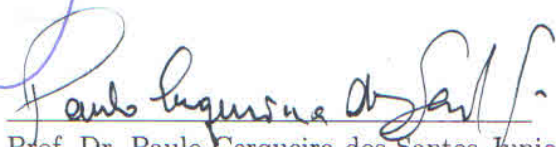
João Marcelo B Protázio

Prof. Dr. João Marcelo Brazão Protázio  
(Coordenador do Programa de Pós-Graduação em Matemática e Estatística – UFPA)

Banca Examinadora

  
Prof. Dr. Héilton Ribeiro Tavares  
PPGME/UFPA  
Orientador

  
Profa. Dra. Maria Regina Madruga Tavares  
PPGME/UFPA  
Coorientadora

  
Prof. Dr. Paulo Cerqueira dos Santos Junior  
Univers. Federal Rural da Amazônia (UFRA)  
Examinador Externo

  
Prof. Dr. Dalton Francisco de Andrade  
Univers. Federal de Santa Catarina (UFSC)  
Examinador Externo

*Aos meus pais e irmãos.*

---

# Agradecimentos

---

Agradeço a Deus, por ter me dado a vida e por tudo que consegui.

Aos meus pais José Miguel e Rosângela, e meus irmãos Marcelo e Miller, por sempre acreditarem em mim e por serem o principal motivo para eu nunca desistir dos meus sonhos.

Agradeço a todos professores do PPGME e, particularmente, aos professores Héilton Tavares, Regina Tavares e Valcir Farias que acreditaram em mim e que contribuíram imensamente para meu crescimento acadêmico e profissional.

A todos os amigos do PPGME, por fornecerem um ambiente de aprendizagem agradável. Agradeço, em especial, aos amigos Alice Moraes, Thamara Medeiros, Fernando Campos, Andrey Nascimento, Eder Lucena, Edney Pantoja, Armando Paiva, Robinson Ortega e Rodrigo Almeida, pelos preciosos momentos de compartilhamento de conhecimento e experiências de vida.

Agradeço ainda, aos meus amigos que participaram do início do meu caminho na vida acadêmica, Alice Moraes, Jhonny Ramos e Frank Dias, vocês foram fundamentais para eu chegar até aqui.

Finalmente, gostaria de agradecer à UFPA pelo ensino gratuito de qualidade, ao LAM, ao PPGME e à CAPES, sem os quais essa dissertação dificilmente poderia ter sido realizada e a todos mais que eu não tenha citado nesta lista de agradecimentos, mas que de uma forma ou de outra contribuíram não apenas para a minha dissertação, mas também para eu ser quem eu sou.

*“A ciência humana de maneira nenhuma nega a existência de Deus. Quando considero quantas e quão maravilhosas coisas o homem compreende, pesquisa e consegue realizar, então reconheço claramente que o espírito humano é obra de Deus, e a mais notável.”*

*Galileu Galilei*

---

# Resumo

---

Este trabalho tem como objetivo apresentar ferramentas de otimização na implementação dos principais métodos para identificar indícios de fraudes em testes, uma área que tem recebido grande importância teórica e em aplicações nos últimos anos. É comum nas avaliações em larga escala a presença de muitos examinados, o que dificulta a aplicação dos métodos de detecção de fraudes em tais avaliações, pois esses métodos se baseiam na comparação entre pares de respostas de indivíduos. Assim, quanto mais indivíduos presentes em um exame, maior será o número de pares a serem analisados e, consequentemente, acarretará em um maior tempo de processamento computacional na detecção de indivíduos que cometeram fraude. Por conta disso, este trabalho explorará a abordagem de processamento em paralelo, diminuindo bastante o tempo computacional das implementações atuais. Além disso, é proposto o uso de duas novas metodologias: a diferença máxima dos escores ( $D_{max}$ ) e o escore mínimo ( $E_{min}$ ). Essas duas propostas foram analisadas e verificou-se que elas contribuem significativamente para a redução do número de pares a serem verificados. Tais técnicas foram implementadas em um pacote construído no ambiente R, denominado *TestFraud*.

**PALAVRAS-CHAVE:** Processamento paralelo, Métodos para detecção de fraude em testes, Avaliação em larga escala, Teoria da Resposta ao Item.



---

# Abstract

---

This work aims to present optimization tools in the implementation of the main methods to identify evidence of fraud in tests, an area that has received great theoretical and application importance in recent years. It is common in the large-scale assessments the presence of many examinees, which makes it hard to apply the methods for detecting cheating, since these are based on the comparison between pairs of individuals' responses. Thus, when the number of individuals increases, more pairs have to be analyzed and, consequently, it will lead to a longer computational processing time in detection of individuals who committed fraud. That is why this work will explore the parallel processing approach, greatly reducing the computational time of the current implementations. In addition, it is proposed to use two new methodologies: the maximum score difference ( $D_{max}$ ) and the minimum score ( $E_{min}$ ). These two proposals were analyzed and it was found their use contribute significantly to reducing the number of pairs to be checked. Such techniques were implemented in a package built in the R environment, denominated *TestFraud*.

**KEYWORDS:** Parallel processing, Methods for detecting cheating on tests, Large scale assessment, Item Response Theory.

---

# Sumário

---

<b>Agradecimentos</b>	<b>vi</b>
<b>Resumo</b>	<b>viii</b>
<b>Abstract</b>	<b>ix</b>
<b>Lista de Tabelas</b>	<b>xii</b>
<b>Lista de Figuras</b>	<b>xiii</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Aspectos gerais . . . . .	1
1.2 Justificativa e importância da dissertação . . . . .	2
1.3 Objetivos . . . . .	2
1.3.1 Objetivo geral . . . . .	2
1.3.2 Objetivos específicos . . . . .	3
1.4 Organização da dissertação . . . . .	3
<b>2 Síntese dos principais métodos da área</b>	<b>5</b>
2.1 Teoria da Resposta ao Item . . . . .	5
2.1.1 Modelo Logístico de 3 parâmetros . . . . .	5
2.1.2 Modelo de Resposta Nominal . . . . .	6
2.2 Métodos de detecção de fraudes . . . . .	7
2.2.1 Índice $\omega$ . . . . .	7
2.2.2 Teste da Binomial Generalizada (GBT) . . . . .	8
2.2.3 Índice $K$ . . . . .	9
2.2.4 Índices $K_1$ e $K_2$ . . . . .	12
2.2.5 Índices $S_1$ e $S_2$ . . . . .	12
<b>3 Processamento preliminar</b>	<b>15</b>
3.1 Escore Verdadeiro . . . . .	15
3.2 Diferença de escores . . . . .	16
3.3 Número mínimo de acertos . . . . .	16
<b>4 Processamento em paralelo no R</b>	<b>18</b>
4.1 Processadores e núcleos . . . . .	18
4.2 Paralelismo no R . . . . .	20

4.2.1 Pacotes <i>doParallel</i> e <i>foreach</i> . . . . .	21
4.3 Comparativos de desempenho . . . . .	22
<b>5 Resultados</b>	<b>24</b>
5.1 Análise das mudanças . . . . .	25
5.2 Estudo da diferença máxima dos escores ( $D_{max}$ ) e do escore mínimo ( $E_{min}$ ) . . . . .	31
5.3 Cenário ENEM 2017 . . . . .	35
<b>6 Conclusões e Considerações Gerais</b>	<b>39</b>
6.1 Trabalhos Futuros . . . . .	40
<b>Referências Bibliográficas</b>	<b>41</b>

---

# Lista de Tabelas

---

5.1	Medidas do tempo de execução em microssegundos da função <i>irtprob</i> usando 100 repetições . . . . .	27
5.2	Medidas do tempo de execução em milissegundos da porção do código utilizada para computação dos índices $K_1$ , $K_2$ , $S_1$ e $S_2$ usando 1.000 repetições	28
5.3	Medidas do tempo de execução em milissegundos das funções <i>similarity2</i> e <i>Fraud.Indices</i> usando 1.000 repetições . . . . .	29
5.4	Medidas do tempo de execução em milissegundos do objeto <i>subgroups</i> usando 100 repetições . . . . .	30
5.5	Medidas do tempo de execução em segundos da função <i>Fraud.Indices</i> aplicada, 10 vezes, para 10.000 pares, com e sem processamento paralelo, utilizando dados simulados com 45 itens e 3.000 indivíduos . . . . .	31
5.6	Medidas do tempo de execução em segundos da função <i>Fraud.Indices</i> aplicada, 1 vez, para 100.000 pares, com e sem processamento paralelo, utilizando dados simulados com 45 itens e 5.000 indivíduos . . . . .	31
5.7	Medidas do tempo de execução em segundos da função <i>Fraud.Indices</i> aplicada, 1 vez, para 100.000 pares, com e sem processamento paralelo, utilizando dados simulados com 45 itens e 15.000 indivíduos . . . . .	31
5.8	Parâmetros $\lambda$ . . . . .	32
5.9	Parâmetros $\zeta$ . . . . .	32
5.10	Medidas dos escores dos indivíduos . . . . .	33
5.11	Probabilidade de não cometer erro Tipo I para $T$ . . . . .	33
5.12	Medidas dos escores dos indivíduos suspeitos . . . . .	34
5.13	Medidas das diferenças de escores dos examinados suspeitos . . . . .	34
5.14	Número de pares a serem analisados após o uso de diferentes valores de $E_{min}$ e $D_{max}$ . . . . .	36
5.15	Percentual do número total de pares a serem analisados após o uso de diferentes valores de $E_{min}$ e $D_{max}$ . . . . .	36
5.16	Tempo de processamento esperado para as 4 áreas utilizando $E_{min} = 30$ e $D_{max} = 5$ , por estado, para o ENEM 2017 . . . . .	37
5.17	Estrutura (número de computadores) necessária para realizar o processo de detecção, por estado, em 15, 30 e 60 dias . . . . .	38

---

# Lista de Figuras

---

4.1	Ilustração de um processador com 4 núcleos . . . . .	19
4.2	Ilustração de um computador com 4 processadores com 4 núcleos cada . . .	19
4.3	Processador Intel Core i7 2600 . . . . .	20
4.4	Ilustração da Lei de Amdahl . . . . .	23
5.1	Funções que calculam probabilidades baseado no MRN no pacote TestFraud e CopyDetect respectivamente . . . . .	26
5.2	Porção do código que computa objetos para obtenção dos índices $K_1, K_2, S_1, S_2$ no pacote Testfraud . . . . .	27
5.3	Porção do código que computa objetos para obtenção dos índices $K_1, K_2, S_1, S_2$ no pacote Copydetect . . . . .	28
5.4	Porção do código referente ao objeto subgroups . . . . .	29
5.5	Porção do código referente ao processamento paralelo utilizando os pacotes DoParallel e foreach . . . . .	30
5.6	Histograma dos escores . . . . .	32
5.7	Histograma dos escores dos indivíduos suspeitos . . . . .	34
5.8	Distribuição da frequência das diferenças de escores dos examinados suspeitos	35

---

# Capítulo 1

## Introdução

---

### 1.1 Aspectos gerais

Os métodos estatísticos para detecção de fraudes em testes têm evoluído muito nos últimos anos [5], mas suas aplicações ainda são extremamente dependentes de recursos computacionais. Uma das maneiras de fraudar um teste é através da *cola*, onde este tipo de fraude consiste em obter respostas por meio de uma destas três formas: anotações; obtenção de respostas de um outro candidato próximo ao examinado; ou por meio de comunicação eletrônica, sendo que esta, por sua vez, possui caráter mais prejudicial devido na maioria dos casos envolver um grupo de estudantes obtendo respostas de uma única fonte, geralmente de alta proficiência. Os métodos de detecção para as duas últimas formas de fraude por *cola* apresentadas tem em comum a comparação de respostas entre pares de indivíduos, com o intuito de detectar uma alta similaridade incomum entre as respostas dos examinados, portanto é notório que o uso desses métodos torna-se muito dispendioso computacionalmente quanto maior for o número de indivíduos presente em um teste, pois quanto mais indivíduos, mais comparações entre respostas serão realizadas.

A presença de um grande número de examinados é recorrente em avaliações educacionais em larga escala. Essas avaliações usam, como instrumentos, testes de proficiência e questionários que permitem avaliar o desempenho escolar e os fatores associados ao mesmo. Os testes são elaborados a partir de Matrizes de Referência. Estas Matrizes indicam o que é avaliado em cada área do conhecimento nas avaliações em larga escala, além de informar as competências e habilidades esperadas em diversos níveis de complexidade, sendo possível aferi-las por meio de testes padronizados [4]. A partir dos resultados obtidos, adquire-se a percepção da qualidade do ensino de uma cidade/estado/país e com isso avaliar onde deve-se intervir para trazer melhores resultados, além de nortear decisões políticas. Sendo assim, este tipo de avaliação é um componente vital para a melhoria da educação nacional.

Uma das principais avaliações nacionais em larga escala é o Exame Nacional do Ensino

Médio (ENEM), destaca-se por ser utilizado como forma parcial ou integral no processo de seleção de estudantes na maioria das universidades públicas. Além disso, é relevante dizer que alguns programas educacionais criados pelo governo brasileiro usam as notas obtidas no ENEM como critério de seleção, tais como: Financiamento Estudantil (FIES) e Programa Universidade para Todos (ProUni), inclusive o Ciências sem Fronteiras (CsF) também utiliza o exame para proporcionar a estudantes de graduação e pós-graduação bolsas de estudo fora do país [6].

Devido a grande importância das avaliações educacionais, em particular o ENEM, é fundamental que os testes sejam confiáveis e precisos a fim de fazer inferências seguras sobre a habilidade de um estudante. Porém, um dos principais fatores que pode invalidar as estimativas é a fraude por *cola*. Este fator transgride a integridade acadêmica e é uma ameaça à qualidade educacional, pois invalida as inferências feitas sobre o conhecimento que se deseja mensurar [11]. Portanto é imprescindível que a detecção desses examinados que cometeram essa transgressão seja realizada em tempo hábil, a fim de retirar esses indivíduos das etapas seguintes sem comprometer os prazos presentes no exame. Sendo assim, torna-se indispensável cada vez mais melhorias na velocidade da computação dos índices responsáveis por essa detecção, sendo crucial a utilização otimizada dos recursos computacionais tal como o processamento em paralelo.

## 1.2 Justificativa e importância da dissertação

As implementações computacionais atuais de métodos para a detecção de fraudes em testes não permitem a utilização dos mesmos nas avaliações em larga escala, devido ao demorado tempo de processamento. Sendo assim, há necessidade de otimizar essas implementações para poder torná-las aplicáveis em grandes avaliações, tal como o ENEM.

## 1.3 Objetivos

### 1.3.1 Objetivo geral

O objetivo principal desta dissertação é o de apresentar e discutir com detalhes a implementação otimizada dos métodos para detecção de potenciais fraudes em testes com a construção do pacote *TestFraud* no R.

### 1.3.2 Objetivos específicos

- i) Apresentar os métodos de detecção de fraude em testes que serão implementados neste trabalho;
- ii) Criar uma função otimizada (*Fraud.Indices*) para cálculo de índices de detecção de fraude para avaliações em larga escala;
- iii) Utilizar processamento paralelo no pacote *TestFraud*;
- iv) Realizar estudos de simulação para avaliar o tempo de processamento na computação dos índices para diferentes cenários de testes;
- v) Realizar estudos de simulação para avaliar o uso da diferença máxima de escore ( $D_{max}$ ) entre pares de indivíduos e escore mínimo ( $E_{min}$ ).

## 1.4 Organização da dissertação

Este trabalho encontra-se dividido em 6 capítulos, a saber:

- No Capítulo 1 é feita uma introdução ao conceito de avaliação em larga escala, em particular o ENEM, e a importância dos métodos de detecção para este exame, são abordados os aspectos gerais, justificativa e importância do trabalho, o objetivo geral e os específicos, e o sumário da dissertação.
- No Capítulo 2 são apresentados os métodos utilizados neste trabalho na área de detecção de fraude.
- No Capítulo 3 são apresentados estudos para redução de pares a serem efetivamente analisados.
- No Capítulo 4 são apresentados os procedimentos utilizados para o processamento em paralelo no R.
- No Capítulo 5 são apresentados resultados de simulação para analisar o desempenho do pacote *TestFraud* em relação ao pacote *CopyDetect*, além disso é apresentado resultados referentes ao uso da diferença máxima de escore ( $D_{max}$ ) entre pares de indivíduos e escore mínimo ( $E_{min}$ ).



- No Capítulo 6 são apresentadas as considerações finais e recomendações para trabalhos futuros.

No próximo capítulo será apresentada a Teoria da Resposta ao Item (TRI), metodologia utilizada para elaboração de testes educacionais mais válidos, fidedignos e precisos na aferição da proficiência de um aluno em determinada área do conhecimento em avaliações em larga escala. Além disso, também será apresentado os principais métodos de detecção de fraudes por *cola*.

---

## Capítulo 2

# Síntese dos principais métodos da área

---

## 2.1 Teoria da Resposta ao Item

Variáveis ou traços latentes são características que não podem ser observadas diretamente, tais como a habilidade de um indivíduo em matemática, o grau de satisfação do consumidor, entre outros. Para inferir sobre os valores destes traços latentes é necessário outras variáveis secundárias que estejam relacionadas a estes traços. A Teoria da Resposta ao Item (TRI) propõe modelos que descrevem a relação entre uma variável latente (ou mais de uma) e a probabilidade associada a algum evento de interesse. Na área educacional, o traço latente que se deseja ser mensurado é a habilidade ou proficiência do estudante (aluno) [1].

Considerando modelos que avaliam apenas um traço latente ou habilidade, os chamados modelos unidimensionais, duas suposições são importantes: a unidimensionalidade e a independência local. A primeira sugere que há apenas uma habilidade responsável pela realização de todos os itens da prova; já a segunda sugere que os itens são respondidos de forma independente por cada indivíduo fixada sua habilidade [1].

### 2.1.1 Modelo Logístico de 3 parâmetros

Para a análise de itens dicotomizados (considerados como sim ou não) são utilizados basicamente 3 (três) tipos de modelos logísticos unidimensionais que se diferem pelo número de parâmetros utilizados [1]:

1. Modelo logístico de 1 parâmetro (dificuldade do item);
2. Modelo logístico de 2 parâmetros, sendo estes: a dificuldade e discriminação;

3. Modelo logístico de 3 parâmetros, sendo estes: a dificuldade, a discriminação e o acerto casual.

Dos três modelos mencionados acima o mais utilizado é o modelo logístico de 3 parâmetros (ML3) e é dado por:

$$P(U_{ij} = 1|\theta_j) = c_i + (1 - c_i) \frac{1}{1 + e^{-Da_i(\theta_j - b_i)}}, \quad (2.1)$$

com  $i = 1, 2, \dots, I$ , e  $j = 1, 2, \dots, n$ , em que:

- $P(U_{ij} = 1|\theta_j)$  é a probabilidade do indivíduo  $j$  com traço latente  $\theta_j$  acertar o item  $i$ ;
- $b_i$  é o parâmetro de dificuldade (ou de posição) do item  $i$ , medido na mesma escala de  $\theta_j$ ;
- $a_i$  é o parâmetro de discriminação (ou inclinação) do item  $i$ , com valor proporcional à inclinação da Curva Característica do Item no ponto  $b_i$ ;
- $c_i$  é o parâmetro de acerto casual do item  $i$ ;
- $D$  é um fator de escala, constante e igual a 1. Utiliza-se o valor 1,702 quando desejar-se que a função logística forneça resultados semelhantes ao da função ogiva normal.

A partir do ML3 é possível obter os demais modelos. Para obter o Modelo logístico de 2 parâmetros basta utilizar  $c_i = 0$ . Já para obter o Modelo logístico de 1 parâmetro faz-se necessário utilizar  $c_i = 0$  e  $a_i = 1$ .

A estimação dos parâmetros  $a_i$ ,  $b_i$ ,  $c_i$  dos itens e do traço latente  $\theta_j$  dos indivíduos é um dos principais objetivos na TRI.

Também ressalta-se o ML3 por ser o modelo utilizado no ENEM para estimar as proficiências dos alunos nas quatro áreas de conhecimento existentes no exame.

### 2.1.2 Modelo de Resposta Nominal

Segundo Andrade et al. [1], o Modelo de Resposta Nominal (MRN), desenvolvido por Bock [3], foi elaborado com o intuito de ter maior precisão da habilidade estimada usando toda a informação contida nas respostas dos indivíduos, e não apenas se o item foi respondido corretamente ou não. Nesse modelo assume-se que a probabilidade de um indivíduo

$j$  selecionar uma particular opção  $v$  (de  $V_i$  opções avaliáveis) do item  $i$  é representada por:

$$P_{iv}(\theta_j) = \frac{e^{(\zeta_{iv} + \lambda_{iv}\theta_j)}}{\sum_{v=1}^{V_i} e^{(\zeta_{iv} + \lambda_{iv}\theta_j)}}, \quad (2.2)$$

com  $i = 1, 2, \dots, I$ ,  $j = 1, 2, \dots, n$ , e  $v = 1, 2, \dots, V_i$ . Em cada  $\theta_j$ , a soma das probabilidades sobre as  $V_i$  opções,  $\sum_{v=1}^{V_i} P_{iv}(\theta_j)$  é 1. As quantidades  $\zeta_{iv}$  e  $\lambda_{iv}$  são parâmetros denominados, respectivamente, de intercepto e inclinação do item para alternativa  $v$  do item  $i$ . Além disso, a estimação dos parâmetros dos itens e as habilidades  $\theta_j$  pode ser realizada pelos métodos de máxima verossimilhança.

## 2.2 Métodos de detecção de fraudes

### 2.2.1 Índice $\omega$

O índice  $\omega$  [16] foi desenvolvido com o objetivo de detectar cópias em testes. Para a construção do índice, considera-se que exista um indivíduo  $c$  suspeito de copiar as respostas do indivíduo  $s$  e que  $h_{cs}$  seja o número de itens que os indivíduos  $c$  e  $s$  responderam na mesma alternativa em um teste de múltipla escolha com opções  $v = 1, \dots, V$ .

A quantidade  $h_{cs}$  condicionada às respostas de  $s$  é definida como

$$h_{cs} = \sum_{i=1}^I 1[u_{ic} = u_{is}], \quad (2.3)$$

com  $i = 1, 2, \dots, I$  representando o  $i$ -ésimo item,  $u_{ic}$  a alternativa do item  $i$  escolhida pelo examinado  $c$ ,  $u_{is}$  a alternativa do item  $i$  escolhida pelo examinado  $s$ , e

$$1[u_{ic} = u_{is}] = \begin{cases} 1, & \text{se } c \text{ e } s \text{ selecionaram a mesma alternativa } v, \\ 0, & \text{c. c.} \end{cases} \quad (2.4)$$

Para a obtenção da distribuição de  $h_{cs}$ , calcula-se a probabilidade de  $c$  selecionar as respostas providas por  $s$ . O valor esperado dessa distribuição é igual a

$$\begin{aligned} E(h_{cs}|\theta_c, U_s, \xi) &= E \left[ \sum_{i=1}^I 1(u_{ic} = u_{is}|\theta_c, U_s, \xi) \right] \\ &= \sum_{i=1}^I E [1(u_{ic} = u_{is}|\theta_c, U_s, \xi)] \\ &= \sum_{i=1}^I [P(u_{ic} = u_{is}|\theta_c, U_s, \xi)], \end{aligned} \quad (2.5)$$

sendo  $\xi$  a matriz de parâmetros dos itens,  $\theta_c$  a habilidade do examinado  $c$ , e  $U_s$  o vetor de respostas do examinado  $s$ .

Assumindo que as respostas dos indivíduos aos itens são localmente independentes, assim como na TRI, a partir das Equações (2.4) e (2.5) condicionando as respostas em  $s$  e os parâmetros dos itens,  $h_{cs}$  é a soma de variáveis Bernoulli independentes cada uma com probabilidade de sucesso, ou seja, com média igual a

$$P(u_{ic} = u_{is} | \theta_c, U_s, \xi), \quad (2.6)$$

sendo assim, o desvio-padrão de  $h_{cs}$  é dado por

$$\sigma_{h_{cs}} = \sqrt{\sum_{i=1}^I [P(u_{ic} = u_{is} | \theta_c, U_s, \xi)][1 - P(u_{ic} = u_{is} | \theta_c, U_s, \xi)]}. \quad (2.7)$$

Para obter  $P(u_{ic} = u_{is} | \theta_c, U_s, \xi)$  usa-se o MRN, descrito na Seção 2.1.2.

Utilizando o Teorema Central do Limite tem-se que

$$\omega = \frac{h_{cs} - E(h_{cs} | \theta_c, U_s, \xi)}{\sigma_{h_{cs}}} \quad (2.8)$$

tem distribuição aproximadamente normal com média 0 e variância 1. A partir disto é possível obter evidências, levando em consideração o valor observado  $\omega$ , que o indivíduo  $c$  cometeu fraude. Para isto basta verificar se o valor observado de  $\omega$  é maior ou igual que o valor crítico para um nível de significância assumido, sendo assim, quanto maior o valor de  $\omega$  mais forte é a evidência de que  $c$  copiou de  $s$  [16].

### 2.2.2 Teste da Binomial Generalizada (GBT)

O Teste da Binomial Generalizada (GBT) utiliza a distribuição binomial composta para analisar o número de respostas idênticas, tanto corretas quanto incorretas, entre dois indivíduos [14]. Seja  $P_{M_i}$  a probabilidade das respostas dos examinados  $c$  e  $s$  ao item  $i$  coincidirem, então  $P_{M_i}$  pode ser calculada como

$$P_{M_i} = \sum_{v=1}^V P_{civ} \cdot P_{siv}, \quad (2.9)$$

sendo  $P_{civ}$  a probabilidade de  $c$  selecionar a alternativa  $v$  do item  $i$  e  $P_{siv}$  a probabilidade do indivíduo  $s$  selecionar a alternativa  $v$  do mesmo item. Além disso, essas probabilidades são computadas de acordo com algum modelo de resposta. Neste trabalho utilizou-se o MRN.

Portanto a probabilidade de serem observadas exatamente  $n$  correspondências (respostas iguais) em  $I$  itens entre dois examinados é igual a

$$f_I(n) = \sum \left( \prod_{i=1}^I P_{M_i}^{u_i} (1 - P_{M_i})^{1-u_i} \right), \quad (2.10)$$

sendo

$$u_i = \begin{cases} 1, & \text{se } c \text{ e } s \text{ respondem identicamente ao item } i, \\ 0, & \text{c.c.} \end{cases} \quad (2.11)$$

e o somatório da Equação (2.10) refere-se a todas possibilidades de combinações de  $n$  respostas coincidentes em  $I$  itens. A partir disso, definiu-se o índice GBT como a cauda superior da distribuição binomial composta e, assim, a probabilidade de observar  $w_{cs} + R_{cs}$  ou mais correspondências em  $I$  itens é igual a

$$\sum_{n=w_{cs}+R_{cs}}^I f_I(n), \quad (2.12)$$

sendo  $w_{cs}$  o número de respostas incorretas coincidentes e  $R_{cs}$  o número de respostas corretas coincidentes [18].

### 2.2.3 Índice $K$

O índice  $K$ , proposto por Holland [7], tem como objetivo avaliar as coincidências de respostas incorretas entre um par de examinados. Para cada par, define-se que existe o examinado suspeito de copiar as respostas, denotado por  $c$ , e o examinado fonte dessas respostas, denotado por  $s$ .

Para esta seção serão utilizadas as seguintes notações:

- $j$ , com  $(j = 1, \dots, J)$ , denotando os examinados;
- $i$ , com  $(i = 1, \dots, I)$ , denotando os itens;
- $v$ , com  $(v = 1, \dots, V)$ , denotando as alternativas de um item;
- $w_j$  sendo o número de respostas incorretas do examinado  $j$ ;
- $r$ , com  $r = 1, \dots, c', \dots, R$ , denotando os subgrupos de examinados, sendo que cada subgrupo tem um número distinto de respostas incorretas,  $R$  é o número total de subgrupos ( $R = I + 1$ , salvo se houver algum subgrupo vazio), além disso, cada

subgrupo possui no mínimo um examinado e que  $\sum_{r=1}^R n_r = J - 1$ , denota-se aqui  $c'$  como o subgrupo ao qual o examinado  $c$  pertence e  $n_r$  é o número total de examinados de cada subgrupo  $r$ ;

- $j'$ , com  $j' = 1, \dots, n_r$ , denotando os examinados dentro de um subgrupo  $r$  específico.
- $\mathbf{M}_r = (M_{r1}, \dots, M_{rj'}, \dots, M_{rn_r})$  sendo um vetor dos números de respostas incorretas idênticas às da fonte em um particular subgrupo  $r$ ;
- $\mathbf{M}_{c'} = (M_{c'1}, \dots, M_{c'n_r})$  denotando o vetor do número de respostas incorretas idênticas às da fonte de  $n_{c'}$  examinados do subgrupo  $c'$ , sendo este o subgrupo que possui o mesmo número de respostas incorretas do copiador.
- $m_{rj'}$  sendo o valor observado do número de respostas incorretas idênticas entre o examinado  $rj'$  e  $s$ ;
- $Q_r = \frac{w_r}{I}$  como a proporção de respostas incorretas de um subgrupo  $r$ , sendo  $w_r$  o número de respostas incorretas do subgrupo  $r$  e  $I$  é o número total de itens do teste.

O índice  $K$  possui duas formas de ser obtido: a partir de uma distribuição amostral empírica ou a partir de uma distribuição teórica.

A construção do índice  $K$  de forma empírica utiliza os dados empíricos de  $J$  examinados respondendo a  $I$  itens. Para esta finalidade tem-se que:

- definir o grupo de examinados com o mesmo número de respostas incorretas de  $c$  (subgrupo  $c'$ );
- para cada examinado do subgrupo  $c'$ , definir o número de itens incorretos idênticos ao examinado  $s$ , obtendo-se assim o vetor  $\mathbf{M}_{c'}$ .

A partir disto, define-se o índice  $K$  como a proporção de examinados com o mesmo número de respostas incorretas que o copiador e cujo número de respostas incorretas correspondentes com a fonte é maior ou igual  $m_{c'c}$ , sendo  $m_{c'c}$  o número de respostas incorretas iguais entre  $c$  e  $s$ . Portanto

$$K = \frac{\sum_{j'=1}^{n_{c'}} I_{c'j'}}{n_{c'}}, \quad (2.13)$$

sendo

$$I_{c'j'} = \begin{cases} 1, & \text{se } m_{c'j'} \geq m_{c'c}, \\ 0, & \text{c.c.} \end{cases}, \quad (2.14)$$

por conseguinte, valores pequenos de  $K$  indicam evidências de que o examinado  $c$  copiou as respostas de  $s$ . Contudo, este método é dependente do tamanho da amostra, pois o número de examinados neste índice pode ser muito pequeno visto que é restrito a um subgrupo particular e isto influencia na precisão do valor do índice  $K$  [12].

Para contornar o problema presente na implementação empírica do índice  $K$ , Holland [7] propôs obter o índice a partir de uma distribuição teórica do número de respostas incorretas iguais entre o examinado  $c'$  e  $s$ , sendo esta variável simplesmente denotada por  $M$ . A construção do índice consiste em utilizar a distribuição binomial para calcular a probabilidade do examinado  $c$  possuir o número de respostas incorretas iguais as do examinado  $s$  maior que os outros examinados pertencentes ao subgrupo  $c'$ . Tem-se portanto que

$$M \stackrel{approx.}{\sim} Bin(w_s, p), \quad (2.15)$$

e o índice  $K$ , representado por  $K^*$ , é então obtido por

$$K^* = P(M \geq m_{c'}) = \sum_{w=m_{c'}}^{w_s} \binom{w_s}{w} (p_{c'}^*)^w (1 - p_{c'}^*)^{w_s - w} \quad (2.16)$$

sendo  $w_s$ , o número de respostas incorretas de  $s$ , o qual é conhecido, e  $p$  é a probabilidade esperada de  $M$ , porém nota-se que  $p$  é desconhecida. Holland [7] propôs estimar  $p$ , denotando esta estimativa por  $p_{c'}^*$ , usando a seguinte expressão

$$p_{c'}^* = \frac{\bar{m}_{c'}}{w_s}, \quad (2.17)$$

sendo

$$\bar{m}_{c'} = \frac{\sum_{j'=1}^{n_{c'}} m_{c'j'}}{n_{c'}}. \quad (2.18)$$

Holland [7] também propôs obter uma estimativa de  $p$  através do método da regressão linear utilizando a proporção de respostas incorretas ( $Q_r$ ) cada subgrupo como a variável preditora. Holland [7] mostrou empiricamente que  $p_r^*$ , sendo  $p_r^*$  definido de modo análogo em 2.17, é linearmente relacionado a  $Q_r$ . Seja  $\hat{p}_r$  a estimativa de  $p_r^*$  usando  $Q_r$ . A expressão para  $\hat{p}_r$  utilizando regressão linear é:

$$\hat{p}_r = \begin{cases} a + bQ_r, & \text{se } 0 < Q_r \leq 0.3; \\ [a + 0.3b] + 0.4b[Q_r - 0.3], & \text{se } 0.3 < Q_r \leq 1. \end{cases} \quad (2.19)$$

Segundo Sotaridona & Meijer [12] observa-se que  $a$  e  $b$  devem ser pré-especificados para o modelo de regressão de duas partes, sendo estas condicionadas ao valor  $Q_r$ . Holland [7]



usou  $a = 0,085$  e diferentes valores para  $b$  baseado na configuração do teste específico utilizado. Porém, não está claro como esses valores foram obtidos no estudo feito por Holland.

### 2.2.4 Índices $K_1$ e $K_2$

Sotaridona & Meijer [12] propuseram estimar  $p_r^*$  através de  $\hat{p}_1^*$  e  $\hat{p}_2^*$ , sendo estes baseados, respectivamente, a partir de uma regressão linear e uma quadrática utilizando  $Q_r$  como variável preditora. Utilizando as estimativas de  $p_r^*$ , duas versões do índice  $K$ ,  $K_1$  e  $K_2$ , são definidas como

$$K_1 = P(M \geq m_{c'c}) = \sum_{w=m_{c'c}}^{w_s} \binom{w_s}{w} (\hat{p}_1^*)^w (1 - \hat{p}_1^*)^{w_s-w} \quad (2.20)$$

e

$$K_2 = P(M \geq m_{c'c}) = \sum_{w=m_{c'c}}^{w_s} \binom{w_s}{w} (\hat{p}_2^*)^w (1 - \hat{p}_2^*)^{w_s-w} \quad (2.21)$$

Vale ressaltar que  $\hat{p}_1^*$  e  $\hat{p}_2^*$  utilizam todos os dados de todos os  $R$  subgrupos para estimar  $p$ , diferente de  $p_{c'}^*$  que utiliza apenas as informações do subgrupo  $c'$  para estimar  $p$ . Sotaridona & Meijer [12] mostraram que  $\hat{p}_2^*$  gerou melhores estimativas para  $p$  do que  $\hat{p}_1^*$  e  $p_{c'}^*$ .

### 2.2.5 Índices $S_1$ e $S_2$

O índice  $S_1$ , proposto por Sotaridona & Meijer [13], é similar aos  $K_1$  e  $K_2$ , uma vez que também é baseado na variável aleatória  $M$ , lembrando que  $M$  representa o número de respostas incorretas iguais entre os examinados  $c'$  e  $s$ . As distinções entres os índices se dá por  $M$  seguir uma distribuição de Poisson, enquanto que os índices  $K_1$  e  $K_2$  atribuem uma distribuição binomial para  $M$ . Além disso, o parâmetro  $\mu$  (a média de  $M$ ) da distribuição de Poisson é estimado a partir de um modelo log-linear. O índice  $S_1$  é representado conforme abaixo:

$$S_1 = P(M \geq m_{c'c}) = \sum_{w=m_{c'c}}^{w_s} \frac{e^{-\hat{\mu}_{c'}} \hat{\mu}_{c'}^w}{w!}, \quad (2.22)$$

sendo  $\hat{\mu}_{c'}$  a estimativa para  $\mu$  utilizando o modelo log-linear, sendo este dado por:

$$\log(\mu_r) = \beta_0 + \beta_1 w_r, \quad \forall r, \quad (2.23)$$

em que  $\beta_0$  e  $\beta_1$  são parâmetros do modelo,  $\mu_r$  é o valor esperado da variável Poisson  $M_{rj'}$  e  $w_r$  é o número de respostas incorretas do subgrupo  $r$ . A partir desse modelo tem-se que  $\hat{\mu}_{c'}$  é dado por

$$\hat{\mu}_{c'} = e^{\beta_0 + \beta_1 w_{c'}}. \quad (2.24)$$

Em contraste com os índices  $K$ ,  $K_1$ ,  $K_2$  e  $S_1$ , o índice  $S_2$  proposto por Sotaridona & Meijer [13] considera tanto as respostas incorretas quanto corretas para a sua computação.

Seja  $M_{rj'}^*$  a soma entre o número de respostas coincidentes incorretas e o número de respostas coincidentes corretas ponderadas entre  $s$  e o examinado  $rj'$  pertencente a um subgrupo  $r$  específico. A expressão  $M_{rj'}^*$  é dada por

$$M_{rj'}^* = M_{rj'} + \sum_{i^*} \delta_{i^* rj'}, \quad (2.25)$$

em que  $\delta_{i^* rj'}$  é a estimativa da informação de cópia do item  $i^*$  pelo examinado  $rj'$ , sendo  $i^*$  representado os itens respondidos corretamente pela fonte. O termo  $\delta_{i^* rj'}$  é definido por:

$$\delta_{i^* rj'} = f(P_{i^* rj'}) = d_1 e^{d_2 P_{i^* rj'}}, \quad (2.26)$$

em que  $0 \leq \delta_{i^* rj'} \leq 1$ , sendo  $P_{i^* rj'}$  a probabilidade do examinado  $rj'$  responder corretamente ao item  $i^*$ . A partir do método da máxima verossimilhança  $P_{i^* rj'}$  é estimado por

$$\hat{P}_{i^* rj'} = \frac{\sum_{j'=1}^{n_r} I_{(u_{i^* rj'} = u_{i^* s})}}{n_r}, \quad (2.27)$$

sendo

$$I_{(u_{i^* rj'} = u_{i^* s})} = \begin{cases} 1, & \text{se } j' \text{ responder corretamente ao item } i^*, \\ 0, & \text{c.c.} \end{cases} \quad (2.28)$$

Os valores  $d_2$  e  $d_1$  são dados por

$$d_2 = - \left( \frac{1+g}{g} \right), \quad (2.29)$$

$$d_1 = - \left( \frac{1+g}{1-g} \right)^{d_2 P_{i^* c}}, \quad (2.30)$$

sendo  $g$  a probabilidade de individuo que desconhece o item acertá-lo ao acaso, ou seja, se um item é composto por  $V$  alternativas então  $g = 1/V$ . [13]

Nota-se que  $M_{rj'}$  se torna um caso especial de  $M_{rj'}^*$  quando não há respostas corretas coincidentes entre  $rj'$  e  $s$ , pois o segundo termo da Equação (2.25) zera. Em contrapartida, quando não há respostas incorretas coincidentes entre  $rj'$  e  $s$  o primeiro termo da Equação

(2.25) zera e  $M_{rj'}^* = \sum_{i^*} \delta_{i^*rj'}$ , tornando-se uma variável sensível para todo conjunto de respostas. Para a aplicação o valor de  $M_{rj'}^*$  é tratado como um número inteiro [13]. Assim o índice  $S_2$  é definido sobre a distribuição de Poisson de  $M_{c'c}^*$  (ou simplesmente  $M^*$ ) e usa o modelo log-linear para estimar média de  $M^*$  assim como é feito para o índice  $S_1$ . O índice  $S_2$  é definido como

$$S_2 = P(M^* \geq m_{c'c}^*) = \sum_{w=m_{c'c}^*}^I \frac{e^{-\hat{\mu}_{c'}} \hat{\mu}_{c'}^w}{w!}, \quad (2.31)$$

sendo  $m_{c'c}^*$  o número observado de coincidências incorretas e corretas ponderada entre os indivíduos  $c$  e  $s$ . Pequenos valores de  $S_2$  indicam que a cópia ocorreu [13].

No próximo capítulo serão apresentados procedimentos para a diminuição do números de pares a serem analisados pelos métodos apresentados nesta seção.

---

## Capítulo 3

# Processamento preliminar

---

Os testes apresentados na Seção 2.2 têm em comum comparar o padrão de resposta de dois indivíduos ( $c$  e  $s$ ). Se o objetivo for analisar todos os indivíduos que realizam um teste, deve-se considerar todas as combinações possíveis de pares, portanto se há  $J$  indivíduos que participam de um teste e deseja-se comparar todos os pares possíveis de respondentes, haverá  $\frac{J(J-1)}{2}$  pares a serem analisados. Considerando um exame como o ENEM, vê-se que é praticamente impossível realizar a análise de todos os pares possíveis em tempo hábil, salvo em supercomputadores, pois vale lembrar que o ENEM apresentou milhões de participantes inscritos nos últimos anos. Deste modo, ao pensar que 1.000.000 de estudantes realizaram o teste, tem-se 499.999.500.000 pares a serem considerados.

As seções seguintes abordam propostas deste trabalho os quais tem por objetivo diminuir a quantidade de pares a serem analisados. Por simplicidade, essas propostas são baseadas em escores, ao invés da proficiência estimada. Mesmo assim, uma quantidade bem importante é o Escore Esperado (ou Escore Verdadeiro) de um indivíduo com habilidade  $\theta$ .

### 3.1 Escore Verdadeiro

Considerando um conjunto de parâmetros de itens  $\zeta_i = (a_i, b_i, c_i)$ ,  $i = 1, \dots, I$  e habilidades  $\theta_j$ , o escore esperado (verdadeiro, ou *true score*, em inglês) de um indivíduo com habilidade  $\theta_j$  é dado por:

$$T_j = \sum_{i=1}^I P(U_{ij} = 1 | \theta_j, \zeta_i) \quad (3.1)$$

## 3.2 Diferença de escores

É notório que a suspeita de fraude por *cola* entre dois respondentes é maior, quando menor for a diferença entre seus escores. Por exemplo, suponha um caso que  $j'$  e  $j$  são dois indivíduos diferentes respondendo a um teste com 45 itens, sendo os seus escores iguais a 35 e 35, respectivamente, e por outro lado suponha outro caso com outros dois indivíduos  $j'$  e  $j$  respondendo ao mesmo teste, porém com escores iguais a 15 e 35, respectivamente. É intuitivo pensar que a suspeita seja maior para o primeiro caso, pois seus escores são iguais, e é muito menor a suspeita para o segundo caso, visto a grande diferença dos seus escores.

Baseado nisso, foi proposto neste trabalho a seguinte metodologia: considere um teste que consiste de  $I$  itens cada um com  $V$  alternativas. Supõe-se  $j'$  e  $j$  dois indivíduos diferentes, e  $E'$  e  $E$  os escores (número total de acertos) de  $j'$  e  $j$  respectivamente, e além disso, considere  $D = |E' - E|$ , e  $\bar{D} = (D_1, D_2, \dots, D_k, \dots, D_K)$ , em que  $D_k$  representa o valor de  $D$  para o  $k$ -ésimo par e  $K$  representa o número total de pares formados. O objetivo é atribuir um valor  $D_{max}$  tal que a partir desse valor pode-se eliminar das análises todos os pares que apresentaram diferença de escores em valor absoluto maior do que  $D_{max}$ .

## 3.3 Número mínimo de acertos

Como dito na Seção 1.1, o ENEM é utilizado como forma de seleção nas principais universidades públicas brasileiras. Com o intuito de conseguir uma vaga nas melhores universidades do Brasil, alguns candidatos usam de meios ilícitos para conseguir tal objetivo. Quadrilhas especializadas em esquemas de fraude fornecem respostas através de um indivíduo (ou mais) de alta proficiência pertencente a quadrilha para candidatos que pagam uma certa quantia para receber as respostas, sendo estas geralmente fornecidas por comunicação eletrônica. Observa-se portanto que é esperado que o número de acertos para esses candidatos seja alto, pois as respostas provém de um individuo de alta proficiência. Sendo assim, é intuitivo que indícios de fraudes recaiam sobre examinados com escores altos. A partir disso foi proposto neste trabalho um valor mínimo do número de acertos ( $E_{min}$ ) tal que abaixo desse valor o examinado não entra no processo de detecção, com isso o número de examinados e, conseqüentemente, o número de pares a serem analisados diminuam.

Estimativas para  $D_{max}$  e  $E_{min}$  dependem fortemente da distribuição dos escores da Teoria Clássica dos Testes (TCT), e, portanto, do número de itens e do comportamento geral da distribuição. Estimativas preliminares podem ser obtidas com base em resultados de simulação, ressaltando que em aplicações reais é fundamental que a etapa preliminar dedique-se a estimar tal distribuição para haver estimativas de  $D_{max}$  e  $E_{min}$  mais apropriadas.

Resultados de simulações relativos a  $D_{max}$  e  $E_{min}$  serão apresentados no Capítulo 5,

---

## Capítulo 4

# Processamento em paralelo no R

---

Com o avanço das aplicações científicas, processos computacionais que utilizam grandes quantidades de dados e cálculos exaustivos sobre eles estão cada vez mais presentes nas mais diversas áreas do conhecimento, tais como: Estatística, Física, Genética, entre outras. Esses processos demandam elevado tempo de processamento, dificultando as pesquisas nas respectivas áreas. Essas aplicações podem obter melhor desempenho se os processos presentes nessas aplicações forem realizados de forma paralela [2].

A programação em paralelo consiste, em termos gerais, em dividir um processamento em partes menores, executar de forma paralela esse processamento e unir o resultado das partes processadas obtendo o resultado de todo o processamento.

Para a realização do processamento em paralelo é essencial que haja uma arquitetura que permita a paralelização do processamento, e além disso, a linguagem de programação utilizada deve permitir tal forma de processamento [2]. Uma linguagem que permite tal tipo de aplicação é a linguagem R.

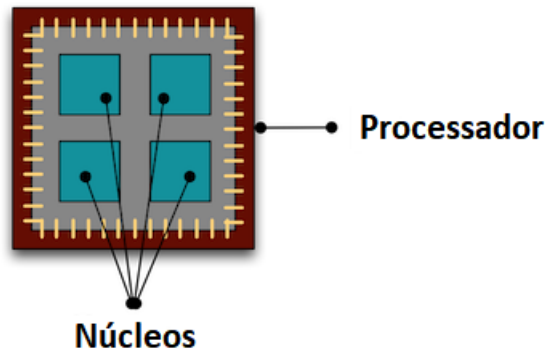
As próximas seções abordam maiores detalhes sobre o processamento em paralelo utilizando a linguagem R.

## 4.1 Processadores e núcleos

O processador, também chamado de microprocessador, CPU (Central Processing Unit) ou UCP (Unidade Central de Processamento) é um chip de silício que processa todas as informações enviadas pelo hardware (memória, HD, placa-mãe e outros) e as operações solicitadas pelo software, se tornando assim uma peça fundamental de um computador. Os computadores da década passada possuíam, geralmente, uma única CPU, mas com o passar dos anos chegamos no período o qual os computadores modernos possuem vários processadores, que, por sua vez, podem conter vários núcleos. Esses processadores e núcleos estão disponíveis para realizar cálculos.

Um computador com um processador pode ter 2 (dual-core), 4 (quad-core) ou mais núcleos, permitindo que cada núcleo execute cálculos ao mesmo tempo, ou seja, cada núcleo funciona como uma unidade de processamento. A Figura 4.1 exibe um processador com 4 núcleos (quad-core).

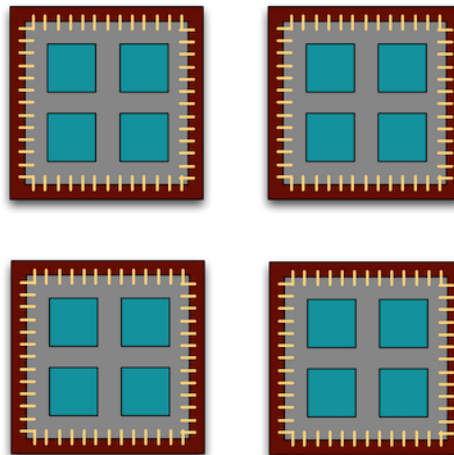
Figura 4.1 *Ilustração de um processador com 4 núcleos*



Fonte: Imagem retirada e adaptada de [8].

Um computador moderno possui múltiplos núcleos e pode possuir mais de um processador. A Figura 4.2 mostra quatro processadores quad-core totalizando 16 núcleos.

Figura 4.2 *Ilustração de um computador com 4 processadores com 4 núcleos cada*



Fonte: Imagem retirada de [8].

Outra tecnologia muito importante no aumento de desempenho dos processadores atuais é a *Simultaneous Multithreading* (SMT). A partir do uso dessa tecnologia, cada núcleo de um processador pode processar duas linhas de execução (*threads*) simultaneamente, reproduzindo a impressão que o número de núcleos do processador dobra, sendo que cada núcleo que é emulado é denominado de núcleo lógico.



A figura a seguir exibe as configurações do processador Intel Core i7 2600. Verifica-se que o processador contém quatro núcleos e tem suporte para trabalhar com até oito *threads*, ou seja, é capaz de emular 8 núcleos (4 físicos e 4 lógicos).

Figura 4.3 *Processador Intel Core i7 2600*

Número do processador	Cache	Velocidade do clock	Nº de núcleos / Nº de threads	Tecnologia Intel® Turbo Boost	Tecnologia de virtualização Intel® (VT-x)	Intel® 64	Intel® Trusted Execution Technology	Gráficos HD Intel®
i7-2600	8.0 MB	3.40 GHz	4 / 8	2.0	✓	✓	✓	✓

Fonte: Site Tecmundo. Acesso 02 de jan de 2019.

Os novos processadores possuem uma quantidade muito maior de núcleos e maior velocidade de processamento. AMD e Intel trazem, respectivamente, como lançamentos os processadores *AMD Threadripper 2990WX* e *Intel Xeon W-3175X*, sendo que ambos suportam a tecnologia SMT. O lançamento da AMD contém 32 núcleos e velocidade de até 4,2 GHz além da capacidade de executar 64 linhas de execução simultaneamente, sendo capaz de emular 64 núcleos. Já o da Intel contém uma quantidade maior de núcleos (28) e velocidade de até 4,3 GHz, além da capacidade de executar 56 linhas de execução simultaneamente, sendo capaz de emular, portanto, 56 núcleos.

## 4.2 Paralelismo no R

Presente nos principais sistemas operacionais, tais como Windows, Linux, Unix e MacOS o R é uma linguagem de programação *open source* e um ambiente voltado principalmente para a computação estatística, modelagem e visualização de dados. Além disso, O R é altamente extensível devido a altíssima disponibilidade de pacotes desenvolvidos pela grande comunidade de utilitários do R.

A linguagem R, por padrão, utiliza apenas uma unidade de processamento, independen-

temente do número de núcleos presentes no(s) processador(es), porém há vários pacotes que proporcionam ao R uma melhor utilização dos recursos computacionais, tal como o processamento em paralelo utilizando vários núcleos. Uma lista atual de pacotes voltados para o processamento em paralelo e computação de alto desempenho encontra-se na página *CRAN Task View: High - Performance and Parallel Computing with R*.

A próxima seção aborda maiores detalhes sobre os pacotes utilizados neste trabalho para a computação em paralelo no R.

### 4.2.1 Pacotes *doParallel* e *foreach*

O pacote *doParallel* é responsável por gerenciar os recursos para utilização de processos em paralelo realizados pelo pacote *foreach*, ou seja, fornece mecanismo necessário para o pacote *foreach* utilizar paralelismo. O pacote *doParallel* age como uma interface entre os pacotes *foreach* e *parallel* [15].

Para começar a utilizar os recursos do pacote *doParallel*, deve-se, primeiramente, definir o grupo de *workspaces* que trabalharão em conjunto para a solução do problema a ser paralelizado, e para criar esse conjunto (grupo de trabalho) utiliza-se a função *makeCluster*, esta tem como argumento principal o número de *workspaces* que serão utilizadas. Por exemplo, se utilizar o comando *makeCluster(4)*, serão criadas 4 *workspaces* que trabalharão em conjunto (em paralelo) para realizar os cálculos de um problema proposto. Em seguida, é necessário um tipo de registro, pois sem tal, a execução ocorrerá normalmente de forma sequencial (não realizando processos em paralelo). Para esse registro, deve-se utilizar a função *registerDoParallel*, ao utilizá-la sem argumentos, no sistema operacional Windows, 3 unidades de processamento são criadas automaticamente, enquanto nos sistemas Unix, metade das unidades de processamento disponíveis no sistema é utilizada. Também pode-se utilizar, como argumento, o objeto retornado pela função *makeCluster*. Por exemplo, ao fazer *cl <- makeCluster(4)* e utilizar *registerDoParallel(cl)* irá registrar que será utilizado um grupo de 4 *workspaces* para trabalharem em conjunto.

Além do *registerDoParallel*, a palavra reservada *%dopar%* diferencia a execução sequencial da paralela no *foreach*, ou seja, ao utilizar a palavra *%dopar%*, iremos fazer com que o laço (*for*) seja feito de forma paralela de acordo com o número de *workspaces* registradas pela função *registerDoParallel*. Outra diferença entre o laço *for* e o do *foreach*, além de sua construção ter a opção de ser paralela, são alguns argumentos que podem ser utilizados no *foreach*. Um dos principais é o “.combine”, que pode gerar a combinação dos resultados

do *foreach* em um objeto nas formas de vetor, soma, multiplicação, matriz ou outras, e caso nada seja declarado, a saída será em formato de lista. Mais sobre o pacote *foreach* pode ser encontrado em [15].

### 4.3 Comparativos de desempenho

Desempenho é a capacidade de reduzir o tempo de resolução do problema à medida que os recursos computacionais aumentam. O *speedup* é uma medida do grau de desempenho e é representada pelo quociente entre o tempo de execução sequencial e o tempo de execução em paralelo [10].

$$S(p) = \frac{T(1)}{T(p)}, \quad (4.1)$$

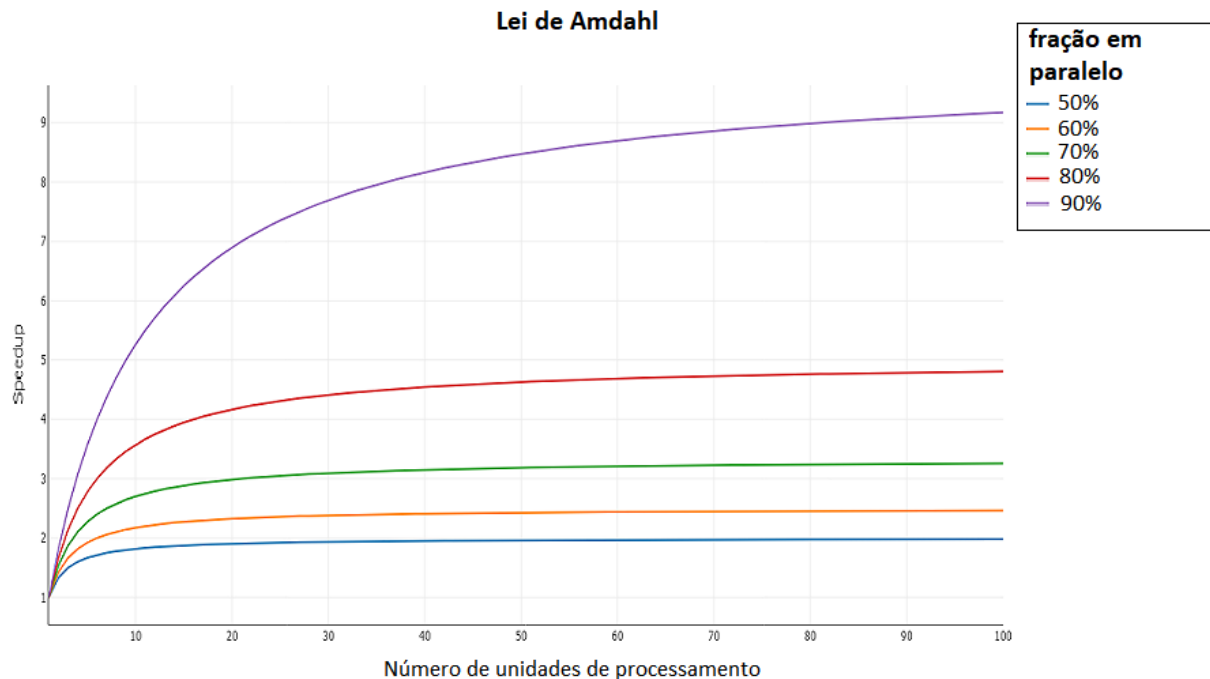
sendo  $T(1)$  o tempo de execução com um processador e  $T(p)$  o tempo de execução com  $p$  processadores. Uma outra medida é a eficiência, esta é uma medida do grau de aproveitamento dos recursos computacionais e é representada pelo quociente entre o grau de desempenho e os recursos computacionais disponíveis:

$$E(p) = \frac{S(p)}{p}. \quad (4.2)$$

Enquanto, em teoria, cada unidade de processamento adicionada aumentaria linearmente o *speedup*, há uma sobrecarga que reduz essa eficiência. Por exemplo, o código e, mais importante, os dados precisam ser copiados para cada CPU adicional, o que demanda tempo. Além disso, novos processos precisam ser criados pelo sistema operacional, o que também leva tempo. Essa sobrecarga reduz a eficiência o suficiente para que os ganhos de desempenho realísticos sejam muito menores do que os teóricos, e geralmente não são redimensionados linearmente como uma função do poder de processamento. Por exemplo, se o tempo de execução de um código é curto, então a sobrecarga (*overhead*) de configurar esses recursos adicionais pode, na verdade, sobrecarregar quaisquer vantagens do poder de processamento adicional. Além disso, nem todas as tarefas podem ser paralelizadas. Dependendo da proporção paralelizada, o *speedup* esperado pode ser significativamente reduzido. A Lei de Amdahl é frequentemente usada para prever o máximo *speedup* teórico usando múltiplos núcleos, em que o *speedup* de um algoritmo é uma função tanto do número de núcleos quanto da proporção do algoritmo que pode ser paralelizada [8]. A Figura 4.4 ilustra um exemplo do funcionamento da Lei de Amdahl, a partir dessa figura

verifica-se que, para este exemplo, o *speedup* considerando 50% do código em paralelo é quase o mesmo utilizando 50 ou 100 unidades de processamento.

Figura 4.4 *Ilustração da Lei de Amdahl*



Fonte: Imagem retirada e adaptada do site Plotly. Acesso 01 de fev de 2019.

No próximo capítulo serão apresentados os resultados de simulações para analisar a redução de tempo utilizando o processamento em paralelo e outras mudanças na execução dos cálculos dos índices de detecção envolvendo muitos pares de examinados.

---

## Capítulo 5

# Resultados

---

Com o intuito de computar índices de similaridade de respostas para testes de múltipla escolha, Zopluoglu [17] desenvolveu o pacote *CopyDetect* utilizando a linguagem R, sendo esse contendo duas funções: uma para computar os índices de similaridade de respostas dicotômicas, denominada de *CopyDetect1* nas primeiras versões, e *similarity1* na versão atual; e outra para computar os índices de similaridade para respostas de múltipla escolha, denominada de *CopyDetect2* nas primeiras versões, e *similarity2* na versão atual. Os índices presentes nas primeiras versões do pacote são  $\omega$ , GBT,  $K$ ,  $K_1$ ,  $K_2$ ,  $S_1$ ,  $S_2$ , e posteriormente incorporou o índice  $M_4$ .

A partir disto, utilizou-se o *CopyDetect* com o objetivo de verificar indícios estatísticos de fraude no Exame Nacional do Ensino Médio (ENEM), contudo percebeu-se que o tempo de processamento para a computação dos índices, considerando apenas poucos indivíduos, foi demoroso, principalmente nas primeiras versões do pacote, sendo assim, não possuindo aplicação prática em testes de larga escala envolvendo milhões de examinados tal como o ENEM.

Além disso, verificou-se que a computação dos índices  $K$ ,  $K_1$ ,  $K_2$ ,  $S_1$  e  $S_2$  continham erros. O indivíduo considerado como fonte ( $s$ ) não foi retirado na obtenção dos subgrupos, sendo assim, não atendendo a condição que  $\sum_{r=1}^R n_r = J - 1$ . A não retirada do indivíduo fonte faz com que a evidência sobre o indivíduo suspeito de colar ( $c$ ) diminua, pois o número de respostas iguais incorretas entre  $s$  e ele próprio ( $s$ ) é máximo (igual ao número de respostas incorretas de  $s$ ), isso faz com que  $p_c^*$ , presente no índice  $K$ , seja maior, impactando no aumento do índice. Esta falha ocorre desde que  $c$  e  $s$  tenham o mesmo número de respostas incorretas, e ainda está presente na versão atual do *CopyDetect*. É notório que isto também afeta a computação dos demais índices ( $K_1$ ,  $K_2$ ,  $S_1$  e  $S_2$ ), diminuindo a evidência sobre  $c$ . Isto se deve ao fato de que os modelos de regressão propostos nesses índices são dependentes das respostas dos indivíduos presentes em cada

subgrupo, e ao comparar o indivíduo  $s$  com ele mesmo faz com que os indícios sobre  $c$  sejam diminuídos. Estas falhas estavam presentes apenas nas primeiras versões do *CopyDetect*, a versão atual encontra-se corrigida.

Perante o exposto, procurou-se corrigir e verificar as fontes do demasiado tempo utilizado no cálculo dos índices. Portanto, várias modificações foram feitas a fim de tornar o processo mais rápido, sendo a computação paralela uma das principais mudanças. As seções a seguir mostram com maiores detalhes as principais diferenças entre as funções que computam os índices de similaridade em respostas de múltipla escolha no *CopyDetect* e no pacote *TestFraud* proposto neste trabalho, sendo a função responsável pelo cálculo dos índices no *TestFraud* denominada de *Fraud.Indices*, além da utilização do processamento em paralelo presente no pacote proposto.

## Máquina de teste

Para todos os resultados obtidos na próxima seção utilizou-se um computador com processador *AMD Ryzen 7 2700*, que possui 8 núcleos físicos com capacidade de executar 16 *threads*, ou seja, possui capacidade de emular 16 núcleos (físicos e lógicos), e opera à frequência de 3.2 Ghz (Max Turbo 4.1 GHz), com 32 GB de memória RAM, Cache L3: 16MB, Cache L2: 4MB, Potência: 65 W. Utilizou-se o sistema operacional Windows 10 Pro 64 bits. Para medir o tempo de execução dos processos utilizou-se o pacote *microbenchmark* [9].

## 5.1 Análise das mudanças

As principais mudanças para aumentar a velocidade do processamento dos índices foram:

1. Otimização na computação de alguns objetos;
2. Diminuição de condições (*if... else...*);
3. Predefinição da dimensão dos objetos;
4. Agrupamento dos cálculos dos índices  $K$ ,  $K_1$ ,  $K_2$ ,  $S_1$ ,  $S_2$ , assim como o agrupamento dos índices  $\omega$  e *GBT*;

5. Argumentos adicionais na função responsável pelo cálculo dos índices;
6. Processamento em paralelo.

A Figura 5.1 apresenta a implementação da função que calcula probabilidades baseada no MRN tendo como argumentos a habilidade do examinado e a matriz dos parâmetros dos itens, sendo as linhas de 1 a 6 representando a função implementada no pacote *TestFraud*, e as linhas de 9 a 20 pertencem ao pacote *CopyDetect*. A partir da Tabela 5.1 verifica-se que, em média, o tempo de execução da função presente no pacote *CopyDetect* é muito maior em relação à implementada no pacote *TestFraud* considerando 100 repetições de execução das funções. O tempo (em microssegundos) de processamento da função do pacote *TestFraud* foi bem menor, principalmente, devido ao fato da não utilização de laços de repetição (*for*), utilizando no lugar operações diretas com matrizes (Figura 5.1, linha 3) e uso de uma função *built-in rowSums* (Figura 5.1, linha 4).

Figura 5.1 Funções que calculam probabilidades baseado no MRN no pacote *TestFraud* e *CopyDetect* respectivamente

```

1- irtprob_TestFraud <- function(ability, item.param) {
2    r <- ncol(item.param)/2
3    ps <- exp((item.param[ ,1:r]*ability)+item.param[ ,1:r+r])
4    prob <- ps/rowSums(ps)
5    prob
6  }
7
8
9- irtprob_CopyDetect <- function(ability, item.param) {
10   prob <- matrix(nrow = nrow(item.param), ncol = ncol(item.param)/2)
11-   for (i in 1:nrow(prob)) {
12     ps <- c()
13-     for (j in 1:ncol(prob)) {
14       ps[j] = exp((item.param[i, j] * ability) +
15                 item.param[i, j + ncol(prob)])
16     }
17     prob[i, ] = ps/sum(ps)
18   }
19   prob
20 }

```

As Figuras 5.2 e 5.3 referem-se a objetos que são computados para obtenção dos índices  $K_1$ ,  $K_2$ ,  $S_1$ ,  $S_2$  nos pacotes *TestFraud* e *CopyDetect* respectivamente. A partir da Tabela 5.2 percebe-se que, em média, o tempo de processamento da computação dos objetos é menor no pacote *TestFraud*. A melhoria no desempenho se deve muito a retirada de transformações nos objetos *smatrix1* (Figura 5.3, linha 12) e *smatrix2* (Figura 5.3, linha

Tabela 5.1 *Medidas do tempo de execução em microssegundos da função irtprob usando 100 repetições*

Pacote	Mín	Q <sub>1</sub>	Média	Mediana	Q <sub>3</sub>	Máx
<i>TestFraud</i>	36,1	38,6	50,3	40,1	41,8	7.423,9
<i>CopyDetect</i>	1.010,7	1.027,4	1.258,0	1.041,9	1.067,4	148.372,0

15) utilizando o comando *as.data.frame*, sendo estas transformações não necessárias para a computação dos índices. Outro fator importante é a retirada de condições (Figura 5.3, linhas 10 e 27), sendo estas substituídas no *TestFraud* por um objeto denominado *pos* (Figura 5.2, linha 3) que identifica as posições que devem ser utilizadas no laço *for*, além da predefinição dos objetos *pr* e *pj* como um vetor de *NA*'s (Figura 5.2, linha 4). Além disso, criou-se um objeto chamado *compare* (Figura 5.2, linha 15) para evitar a comparação de matrizes mais de uma vez como ocorre no pacote *CopyDetect* (Figura 5.3, linhas 21 e 24). Percebe-se também que a computação do objeto *incorrect.items* aparece 2 duas vezes (Figura 5.3, linhas 2 e 11) sendo desnecessária a repetição.

Figura 5.2 *Porção do código que computa objetos para obtenção dos índices K<sub>1</sub>, K<sub>2</sub>, S<sub>1</sub>, S<sub>2</sub> no pacote Testfraud*

```

1 ws <- sum(form2[pa[2], ] == 0, na.rm = TRUE)
2 incorrect.items <- which(form2[pa[2], ] == 0)
3 pos <- which(lengths!=0)
4 pr <- pj <- rep(NA, (I+1))
5 prob <- weight <- matrix(nrow=(I+1), ncol=I)
6 g <- 1/length(options)
7 d2 <- -(1+g)/g
8 p1 <- ((1+g)/(1-g))*exp(1)
9 for(j in pos){
10   smatrix1 <- matrix(rep(as.matrix(form[pa[2], incorrect.items]),
11     lengths[j]), nrow=lengths[j], byrow=TRUE)
12   smatrix2 <- matrix(rep(as.matrix(form2[pa[2], ]), lengths[j]),
13     nrow=lengths[j], byrow=TRUE)
14   pr[j] <- mean(rowSums(form[subgroups[[j]], incorrect.items]==smatrix1))/ws
15   compare <- (form2[subgroups[[j]], ]==1)&(smatrix2==1)
16   prob[j,] <- colMeans(compare)
17   weight[j,] <- p1^(prob[j,]*d2)
18   pj[j] <- mean(((compare)*1)%%as.matrix(weight[j, ]))
19 }

```

A Tabela 5.3 apresenta medidas do tempo (em milissegundos) de execução, utilizando 1.000 repetições, das funções *similarity2* e *Fraud.Indices* aplicadas apenas para as computações dos índices, ou seja, adaptou-se a função *similarity2* para computar apenas



Figura 5.3 Porção do código que computa objetos para obtenção dos índices  $K_1$ ,  $K_2$ ,  $S_1$ ,  $S_2$  no pacote *Copydetect*

```

1 ws <- sum(form2[pa[2], ] == 0, na.rm = TRUE)
2 incorrect.items <- which(form2[pa[2], ] == 0)
3 pr <- c()
4 prob <- matrix(nrow = (ncol(form) + 1), ncol = ncol(form))
5 weight <- matrix(nrow = (ncol(form) + 1), ncol = ncol(form))
6 pj <- c()
7 g = 1/length(resp.options)
8 d2 = -(1 + g)/g
9- for (j in 1:(ncol(form) + 1)) {
10-   if (length(subgroups[[j]]) != 0) {
11     incorrect.items <- which(form2[pa[2], ] == 0)
12     smatrix1 <- as.data.frame(matrix(rep(as.matrix(form[pa[2],
13                                       incorrect.items]), length(subgroups[[j]])),
14                                   nrow = length(subgroups[[j]]), byrow = TRUE))
15     smatrix2 <- as.data.frame(matrix(rep(as.matrix(form2[pa[2], ]),
16                                       length(subgroups[[j]])),
17                                   nrow = length(subgroups[[j]]), byrow = TRUE))
18     emp.agg <- rowSums(form[subgroups[[j]], incorrect.items] == smatrix1,
19                       na.rm = TRUE)
20     pr[j] = mean(emp.agg, na.rm = TRUE)/ws
21     prob[j, ] <- colMeans((form2[subgroups[[j]], ] == 1) & (smatrix2 == 1),
22                          na.rm = TRUE)
23     weight[j, ] <- (((1 + g)/(1 - g)) * exp(1))^(prob[j, ] * d2)
24     pj[j] <- mean(((form2[subgroups[[j]], ] == 1 &
25                  smatrix2 == 1) * 1) %>% t(t(weight[j, ])), na.rm = TRUE)
26   }
27-   else if (length(subgroups[[j]]) == 0) {
28     pr[j] = NA
29     pj[j] = NA
30   }
31 }

```

Tabela 5.2 Medidas do tempo de execução em milissegundos da porção do código utilizada para computação dos índices  $K_1$ ,  $K_2$ ,  $S_1$  e  $S_2$  usando 1.000 repetições

Pacote	Mín	$Q_1$	Média	Mediana	$Q_3$	Máx
<i>TestFraud</i>	158,1	161,1	187,4	165,4	174,0	1.107,7
<i>CopyDetect</i>	360,8	374,5	437,1	387,0	529,1	1.323,6

os índices, pois esta função gera um relatório após a sua execução. Portanto, retirou-se a geração desse relatório para poder comparar efetivamente apenas as computações do índices, assim como é na função *Fraud.Indices*. Observa-se pela Tabela 5.3 que a função *Fraud.Indices* possui, em média, menor tempo de processamento (188,7 ms), percebe-se, na Tabela 5.2, que a melhoria se dá principalmente pela otimização na criação dos objetos presentes na Figura 5.2. Além disso, juntou-se a computação dos índices  $K$ ,  $K_1$ ,  $K_2$ ,  $S_1$ ,

$S_2$  em uma função interna, assim como a dos índices  $\omega$  e  $GBT$  em uma outra função interna, esse agrupamento ocorreu pelo fato que esses índices utilizam alguns objetos iguais, então evitou-se a computação de um objeto específico mais de uma vez, pois na função `similarity2` há 4 funções internas para a computação dos índices, sendo uma para o índice  $\omega$ , uma para o  $GBT$ , uma para o índice  $K$ , e outra para os índices  $K_1$ ,  $K_2$ ,  $S_1$  e  $S_2$ . Sendo assim ocorria repetição no cálculo de um mesmo objeto, por exemplo,  $\omega$  e  $GBT$  necessitam do número e respostas coincidentes (corretas e incorretas) entre  $c$  e  $s$ , de forma que ao criar funções separadas esses objetos são computados mais de uma vez. Todas modificações fazem com que o tempo de processamento da função `Fraud.Indices` seja menor (Tabela 5.3).

Tabela 5.3 *Medidas do tempo de execução em milissegundos das funções `similarity2` e `Fraud.Indices` usando 1.000 repetições*

Função	Mín	$Q_1$	Média	Mediana	$Q_3$	Máx
<code>Fraud.Indices</code>	161,9	164,7	188,7	168,1	175,9	1.115,1
<code>similarity2</code>	551,9	566,3	655,8	706,2	729,1	1.540,4

Um outro fator importante é que ao executar a função `similarity2` para mais de um par de respondentes, um objeto denominado `subgroups` é computado para cada par, mas esse objeto é o mesmo para cada execução. Portanto, utilizou-se um argumento adicional na função `Fraud.Indices` próprio para receber o objeto `subgroups`, e assim ele é executado apenas uma vez, e não para cada par, como é na função `similarity2`. A criação desse objeto (Figura 5.4) demanda grande tempo computacional, e este cresce à medida que o número de respondentes aumenta (Tabela 5.4).

Figura 5.4 *Porção do código referente ao objeto `subgroups`*

```

1 subgroups <- vector("list", (ncol(scored.data)+1))
2 for(j in 1:(ncol(scored.data)+1)){
3   subgroups[[j]] <- which((ncol(scored.data)-rowSums(scored.data,
4                                     na.rm= T))==j-1)
5 }

```

Para diminuir mais o processamento dos cálculos dos índices envolvendo muitos pares, utilizou-se programação em paralelo (Figura 5.5) conforme apresentado na Seção 4.2.1. Observa-se também o uso da função `detectCores` (Figura 5.5, linha 2), que retorna o número de núcleos presentes no sistema (físicos e lógicos); como o computador utilizado

Tabela 5.4 Medidas do tempo de execução em milissegundos do objeto *subgroups* usando 100 repetições

N	Mín	Q <sub>1</sub>	Média	Mediana	Q <sub>3</sub>	Máx
3.000	54,6	55,2	69,5	60,0	60,9	258,6
5.000	76,8	78,0	100,0	83,9	86,0	264,3
15.000	191,8	207,0	280,6	215,5	386,0	1.173,5

possui 16, usou-se 15 núcleos, deixando um livre para o sistema Windows. Além disso, utilizou-se o argumento “.combine” igual a *rbind* (Figura 5.5, linha 8), este argumento é estritamente necessário, pois para cada iteração é gerado um vetor com 7 elementos, e com o uso do *rbind* tem-se que cada vetor comporá uma linha de uma matriz, e o objeto gerado pelo *foreach* será uma matriz com o número de linhas igual ao número de pares a serem analisados, e o número de colunas igual a 7 (número de índices).

Figura 5.5 Porção do código referente ao processamento paralelo utilizando os pacotes *DoParallel* e *foreach*

```

1 library(doParallel)
2 c1=makeCluster(detectCores()-1)
3 registerDoParallel(c1)
4 #####
5 #carregar pacotes que são utilizados
6 #Objetos criados (responses, ipar e outros)
7 #####
8 pairs[,3:10]=foreach(i=1:nrow(pairs),.combine = rbind)%dopar%{
9   Fraud.Indices(data=responses, item.par=ipar,
10                pair=c(pairs[i,1],pairs[i,2]),
11                options=c("A","B","C","D","E"),
12                key=gab, scored.data=scored.data,
13                abilities=abilities,
14                wc = wc[pairs[i,1]],
15                subgroup = subgroups[[wc[pairs[i,1]]+1]],
16                subgroups=subgroups,lengths=lengths)
17 }
18 stopCluster(c1)

```

A partir da Tabela 5.5 verifica-se, considerando a média, que o *speedup* utilizando 15 núcleos foi de 5,66, ou seja, 5,66 vezes mais rápido que o processamento sequencial, apresentando uma eficiência de 0,38 para um cenário de um teste composto por 45 itens, 3.000 participantes e 10.000 pares. O *speedup* e eficiência pouco mudaram para diferentes cenários (Tabelas 5.6 e 5.7). Para um cenário com 45 itens, 5.000 participantes e 100.000 pares (Tabela 5.6) o *speedup* foi de 5,48 apresentando uma eficiência de 0,37. Já para

um cenário com 45 itens, 15.000 participantes e 100.000 pares (Tabela 5.7) o *speedup* e a eficiência foram, respectivamente, 5,65 e 0,38.

Tabela 5.5 *Medidas do tempo de execução em segundos da função Fraud.Indices aplicada, 10 vezes, para 10.000 pares, com e sem processamento paralelo, utilizando dados simulados com 45 itens e 3.000 indivíduos*

Estrutura	Mín	$Q_1$	Média	Mediana	$Q_3$	Máx
<i>foreach</i>	238,5	239,0	243,3	243,0	247,7	248,2
<i>for</i>	1.371,9	1.375,8	1.377,6	1.376,7	1.379,5	1.385,0

Tabela 5.6 *Medidas do tempo de execução em segundos da função Fraud.Indices aplicada, 1 vez, para 100.000 pares, com e sem processamento paralelo, utilizando dados simulados com 45 itens e 5.000 indivíduos*

Estrutura	Mín	$Q_1$	Média	Mediana	$Q_3$	Máx
<i>foreach</i>	2.528,4	2.528,4	2.528,4	2.528,4	2.528,4	2.528,4
<i>for</i>	13.845,5	13.845,5	13.845,5	13.845,5	13.845,5	13.845,5

Tabela 5.7 *Medidas do tempo de execução em segundos da função Fraud.Indices aplicada, 1 vez, para 100.000 pares, com e sem processamento paralelo, utilizando dados simulados com 45 itens e 15.000 indivíduos*

Estrutura	Mín	$Q_1$	Média	Mediana	$Q_3$	Máx
<i>foreach</i>	2.643,5	2.643,5	2.643,5	2.643,5	2.643,5	2.643,5
<i>for</i>	14.948,0	14.948,0	14.948,0	14.948,0	14.948,0	14.948,0

## 5.2 Estudo da diferença máxima dos escores ( $D_{max}$ ) e do escore mínimo ( $E_{min}$ )

Com o objetivo de analisar as propostas apresentadas no Capítulo 3 fez-se o uso de dados simulados tendo como cenário 2.000 indivíduos respondendo a 45 itens com 5 alternativas cada item, sendo as habilidades dos indivíduos simuladas a partir de uma  $N(0, 1)$  e os parâmetros dos itens do MRN a partir das seguintes distribuições:  $\lambda_{iv} \sim N(0, 1)$  e  $\zeta_{iv} = 2\lambda_{iv} + X$ , sendo  $X \sim N(0, 10^{-2})$ . As Tabelas 5.8 e 5.9 contêm os parâmetros dos 5 primeiros itens dos 45 simulados.

Tabela 5.8 *Parâmetros  $\lambda$*

Item ( $i$ )	$\lambda_1$	$\lambda_2$	$\lambda_3$	$\lambda_4$	$\lambda_5$
1	-1,03	0,99	0,72	0,44	-1,12
2	-0,06	-0,40	0,99	0,54	-1,07
3	0,90	-0,27	-1,04	0,34	0,07
4	-0,55	0,11	-0,71	1,71	-0,55
5	0,46	-0,32	-0,54	-0,50	0,90

Tabela 5.9 *Parâmetros  $\zeta$*

Item ( $i$ )	$\zeta_1$	$\zeta_2$	$\zeta_3$	$\zeta_4$	$\zeta_5$
1	-2,00	1,72	1,50	1,04	-2,26
2	-0,16	-0,84	2,07	1,18	-2,25
3	1,93	-0,51	-2,03	0,68	-0,08
4	-1,13	0,11	-1,40	3,40	-0,98
5	1,14	-0,63	-1,11	-1,20	1,79

As propostas do Capítulo 3 baseiam-se nos escores dos indivíduos. Sendo assim, mostra-se através da Figura 5.6 a distribuição dos escores dos indivíduos, e verifica-se que a distribuição apresentou uma leve assimetria negativa, tendo como média 25,34. A partir da Tabela 5.10 verifica-se que o menor escore foi 1 e o maior escore foi 40.

Figura 5.6 *Histograma dos escores*

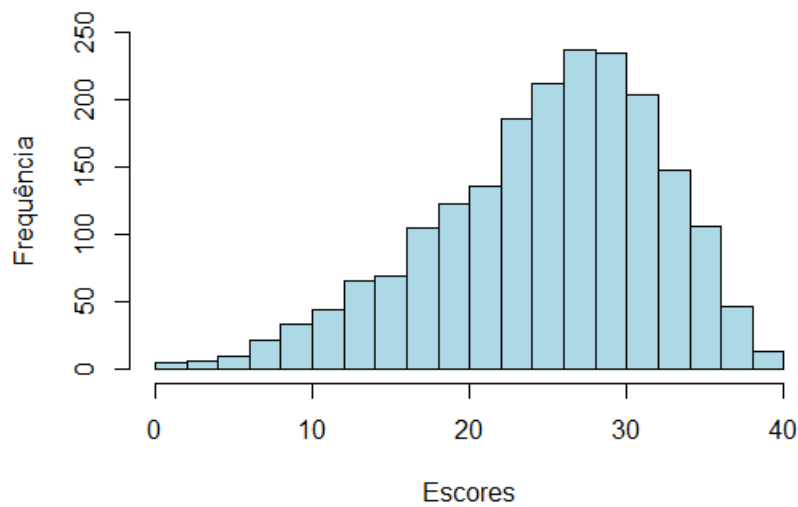


Tabela 5.10 *Medidas dos escores dos indivíduos*

Mín	$Q_1$	Média	Mediana	$Q_3$	Máx
1,00	21,00	25,34	26,00	31,00	40,00

Para controle do erro Tipo I (probabilidade de indicar fraude quando na verdade não ocorreu), utilizou-se uma variável  $T$  que representa o número mínimo de testes que indicaram indícios de fraude, ou seja, serão considerados suspeitos de fraude os pares detectados em  $t$  índices ou mais. Para as análises a seguir considerou-se que um par é considerado suspeito se pelo menos 4 índices (opção conservadora) indicarem indícios de fraude, ou seja, se pelo menos 4 índices apresentarem  $p$ -valor menor que o nível de significância ( $\alpha$ ) assumido usado para cada um dos 7 índices. Para os resultados a seguir utilizou-se  $\alpha = 0,001$ .

Tabela 5.11 *Probabilidade de não cometer erro Tipo I para  $T$*

$\alpha$	$T$						
	1	2	3	4	5	6	7
0,001	0,99841	0,99958	0,99987	0,99994	0,99996	0,99998	0,99999
0,005	0,99200	0,99714	0,99895	0,99932	0,99961	0,99981	0,99992
0,010	0,98413	0,99347	0,99732	0,99815	0,99883	0,99942	0,99977
0,020	0,96841	0,98501	0,99312	0,99498	0,99659	0,99822	0,99920
0,050	0,92146	0,95489	0,97646	0,98162	0,98596	0,99218	0,99585

Vale ressaltar que o valor de  $\alpha$  estipulado não necessariamente deve coincidir com a coluna  $T = 1$ , pois ele foi usado individualmente para os índices, e pode não refletir na distribuição de  $T$  da mesma forma. Pode-se, de fato, notar através da Tabela 5.11 que a coluna  $T = 2$  é a que fica mais próxima do real valor de  $\alpha$ . No entanto, será adotada uma decisão mais conservadora, em princípio, com o evento  $\{T = 4\}$  para indicar suspeita acentuada de fraude.

A Figura 5.7 apresenta a distribuição dos escores dos indivíduos considerados suspeitos. A partir dessa figura e também da Tabela 5.12, verifica-se que há possibilidade de detectar fraude para diferentes níveis de escore.

A Figura 5.8 apresenta a distribuição da diferença dos escores entre os pares de indivíduos detectados. A partir dessa distribuição verifica-se que há possibilidade de detectar um par de indivíduos com alta diferença de escore. Com base na Tabela 5.13 verifica-se que

Figura 5.7 *Histograma dos escores dos indivíduos suspeitos*

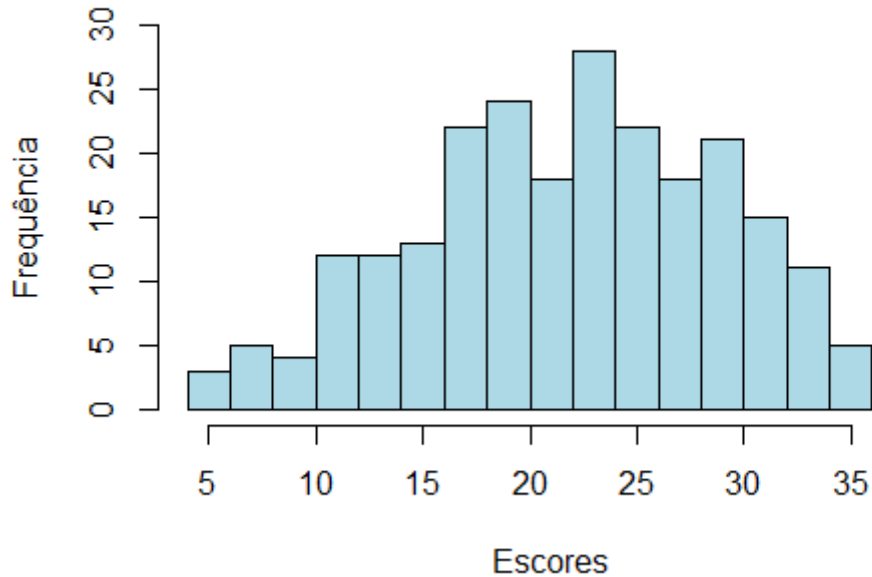


Tabela 5.12 *Medidas dos escores dos indivíduos suspeitos*

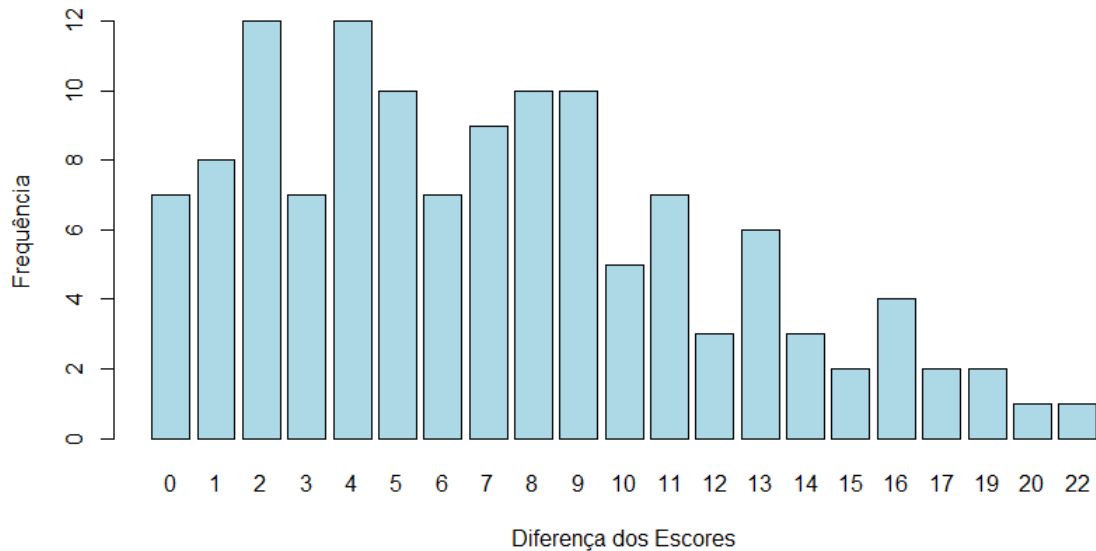
Mín	$Q_1$	Média	Mediana	$Q_3$	Máx
4,00	17,00	22,23	23,00	28,00	36,00

50% dos pares detectados apresentaram diferença de escore menor ou igual a 7, também verifica-se que a maior diferença foi de 22.

Tabela 5.13 *Medidas das diferenças de escores dos examinados suspeitos*

Mín	$Q_1$	Média	Mediana	$Q_3$	Máx
0,00	3,00	7,16	7,00	10,00	22,00

Os resultados apresentados mostraram que mesmo indivíduos com escores baixos e pares com diferenças altas de escore podem ser detectados, porém deve-se lembrar que quando trata-se de esquemas de fraude é comum que as respostas fornecidas pela fonte são dadas integralmente, ou próximo disso, para os examinados pertencentes ao esquema. Além disso, como foi dito na Seção 3.3, as respostas provêm de um indivíduo (ou mais) de

Figura 5.8 *Distribuição da frequência das diferenças de escores dos examinados suspeitos*

alta proficiência, logo é esperado que este indivíduo possua um escore alto, assim como os demais que recebem as respostas do indivíduo fonte. Sendo assim, verificou-se o impacto na redução no número de pares (Tabelas 5.14 5.15) a serem analisados para diferentes valores de escore mínimo ( $E_{min}$ ) e diferença máxima de escore ( $D_{max}$ ).

Percebe-se que a partir da Tabela 5.14 usando  $D_{max} = 20$  e  $E_{min} = 10$  o número de pares total a serem analisados caiu de 1.999.000 ( $C_{2000,2}$ ) para 1.838.652, ou seja, 91,98% do número total de pares seriam analisados. Já ao considerar  $D_{max} = 4$  e  $E_{min} = 30$  verifica-se que apenas 8,10% do número total de pares seriam analisados. Portanto, o uso do  $E_{min}$  e  $D_{max}$  gera uma considerável redução no número de pares a serem analisados, e consequentemente ocorrerá uma redução no tempo de processamento para a identificação de pares suspeitos.

### 5.3 Cenário ENEM 2017

Para analisar um cenário como o presente no ENEM, realizou-se outra simulação para verificar o efeito do  $E_{min}$  e  $D_{max}$  em uma base com uma quantidade maior de indivíduos. Para isso, utilizou-se os mesmos parâmetros de itens simulados presentes na seção anterior, porém utilizando 10.000 examinados. A partir da simulação verificou-se que o percentual



Tabela 5.14 *Número de pares a serem analisados após o uso de diferentes valores de  $E_{min}$  e  $D_{max}$* 

$D_{max}$	$E_{min}$					
	10	14	18	22	26	30
20	1.838.652	1.691.097	1.434.719	1.056.331	605.550	201.930
16	1.734.787	1.628.915	1.413.478	1.055.077	605.550	201.930
12	1.529.697	1.465.550	1.317.196	1.024.942	603.632	201.930
8	1.189.877	1.155.401	1.073.316	887.928	562.252	199.978
4	694.965	681.004	645.744	565.028	395.423	161.903

Tabela 5.15 *Percentual do número total de pares a serem analisados após o uso de diferentes valores de  $E_{min}$  e  $D_{max}$* 

$D_{max}$	$E_{min}$					
	10	14	18	22	26	30
20	91,98	84,60	71,77	52,84	30,29	10,10
16	86,78	81,49	70,71	52,78	30,29	10,10
12	76,52	73,31	65,89	51,27	30,20	10,10
8	59,52	57,80	53,69	44,42	28,13	10,00
4	34,77	34,07	32,30	28,27	19,78	8,10

do total de pares gerados (49.995.000) utilizando como filtro  $E_{min} = 30$  e  $D_{max} = 5$  foi de 8,58%, ou seja, o número de pares a serem analisados caiu para 4.287.252. Utilizando desse percentual gerou-se a Tabela 5.16 que apresenta o tempo esperado de execução (em dias) do cálculo dos índices para as 4 áreas de conhecimento do ENEM, por estado, utilizando como filtro  $E_{min} = 30$  e  $D_{max} = 5$ . Além disso, a tabela exhibe o número de participantes do ENEM 2017, por estado, que responderam questões relacionadas às 4 áreas do ENEM; o número de pares total gerado, por estado; e o número esperado de pares após o filtro, por estado. Para a estimativa do tempo utilizou-se o tempo de processamento esperado por par que pode ser obtido pela Tabela 5.7, bastando fazer o quociente entre o tempo de processamento e o número de pares analisados, ou seja,  $\frac{2.643,5}{100.000} = 0,026$  segundos. Pela Tabela 5.16 verifica-se que o menor tempo de processamento esperado ocorre no estado de Roraima (7 dias), já o maior é no estado de São Paulo (27.391 dias).

A Tabela 5.17 contém informações sobre a quantidade de computadores, com as mesmas configurações descritas no início deste capítulo, que seriam necessários para retornar os resultados do processo de detecção, por estado, em 15, 30 e 60 dias. Verifica-se que o

Tabela 5.16 *Tempo de processamento esperado para as 4 áreas utilizando  $E_{min} = 30$  e  $D_{max} = 5$ , por estado, para o ENEM 2017*

Estado	Frequência	Número total de pares	Número de pares esperados para $E_{min} = 30$ e $D_{max} = 5$	Tempo de processamento esperado (em dias)
AC	34.580	597.870.910	51.297.324	62
AL	82.984	3.443.130.636	295.420.609	356
AM	90.081	4.057.248.240	348.111.899	419
AP	31.673	501.573.628	43.035.017	52
BA	313.898	49.265.820.253	4.227.007.378	5.088
CE	252.344	31.838.620.996	2.731.753.681	3.288
DF	82.517	3.404.486.386	292.104.932	352
ES	91.238	4.162.140.703	357.111.672	430
GO	143.007	10.225.429.521	877.341.853	1.056
MA	195.357	19.082.081.046	1.637.242.554	1.971
MG	483.133	116.708.506.278	10.013.589.839	12.053
MS	54.482	1.484.116.921	127.337.232	153
MT	75.566	2.855.072.395	244.965.211	295
PA	239.550	28.691.981.475	2.461.772.011	2.963
PB	134.505	9.045.730.260	776.123.656	934
PE	247.691	30.675.291.895	2.631.940.045	3.168
PI	105.332	5.547.362.446	475.963.698	573
PR	189.714	17.995.606.041	1.544.022.998	1.859
RJ	290.230	42.116.581.335	3.613.602.679	4.350
RN	109.455	5.990.143.785	513.954.337	619
RO	51.111	1.306.141.605	112.066.950	135
RR	11.724	68.720.226	5.896.195	7
RS	195.823	19.173.225.753	1.645.062.770	1.980
SC	85.792	3.680.090.736	315.751.785	380
SE	62.788	1.971.135.078	169.123.390	204
SP	728.310	265.217.363.895	22.755.649.822	27.391
TO	45.878	1.052.372.503	90.293.561	109

estado de São Paulo requer uma maior estrutura (em relação a número de computadores), necessitando de 1826 computadores para realizar o processo de detecção em 15 dias. Já os estados de Roraima, Amapá e Acre são os que requerem menor estrutura para realizar o processo de detecção em 15 dias com 1, 3 e 4 computadores respectivamente.

Tabela 5.17 *Estrutura (número de computadores) necessária para realizar o processo de detecção, por estado, em 15, 30 e 60 dias*

Estado	Número de computadores (15 dias)	Número de computadores (30 dias)	Número de computadores (60 dias)
AC	4	2	1
AL	24	12	6
AM	28	14	7
AP	3	2	1
BA	339	170	85
CE	219	110	55
DF	23	12	6
ES	29	14	7
GO	70	35	18
MA	131	66	33
MG	804	402	201
MS	10	5	3
MT	20	10	5
PA	198	99	49
PB	62	31	16
PE	211	106	53
PI	38	19	10
PR	124	62	31
RJ	290	145	72
RN	41	21	10
RO	9	4	2
RR	1	1	1
RS	132	66	33
SC	25	13	6
SE	14	7	3
SP	1826	913	457
TO	7	4	2

---

## Capítulo 6

# Conclusões e Considerações Gerais

---

Neste estudo foram apresentados sete métodos estatísticos de detecção de fraudes em testes, bem como apresentadas as dificuldades na aplicação desses métodos nas avaliações em larga escala. A principal dificuldade é o tempo computacional envolvido no cálculo dos índices dos métodos apresentados quando há muitos participantes em uma avaliação. Assim, este trabalho apresentou propostas para tornar os métodos apresentados aplicáveis a esse tipo de avaliação.

Duas novas propostas foram avaliadas com o objetivo de diminuir a quantidade de pares a serem analisados: o escore mínimo do candidato ( $E_{min}$ ) e a diferença de escore máxima entre os pares ( $D_{max}$ ). Através de um estudo envolvendo dados simulados mostrou-se que os índices podem detectar indícios de fraude para diferentes níveis de escore e também para diferenças altas de escore entre pares de indivíduos. Também avaliou-se como se dá a redução no número de pares a serem analisados utilizando diferentes valores de  $E_{min}$  e  $D_{max}$ . A utilização do  $E_{min}$  e  $D_{max}$  é útil quando trata-se de detecção de esquemas de fraude, pois é comum nestes casos que o indivíduo fonte forneça integralmente, ou quase integralmente, as respostas para os copiadores. Além disso, a fonte geralmente possui alta proficiência e, por consequência, um escore alto, o que vai refletir também em um escore alto dos indivíduos copiadores.

Foi apresentada também a construção de um pacote otimizado para a detecção de fraudes em testes denominado *TestFraud*, inicialmente baseado no pacote *CopyDetect* [17]. No entanto, no pacote *TestFraud* muitas funções foram otimizadas e também utilizou-se a programação em paralelo com o objetivo de reduzir o tempo de processamento na computação dos índices para um grande número de pares de indivíduos. Através dos resultados observou-se, de modo geral, melhoras significativas em relação à implementação presente no pacote *CopyDetect*, e a melhora se acentuou quando a implementação do cálculo dos índices para um grande número de pares foi realizada utilizando processamento

paralelo, ressaltando assim a importância do uso deste tipo de processamento na área de detecção de fraude.

Porém, ao analisar um cenário como o do ENEM, percebeu-se que ainda são necessárias melhorias na velocidade da computação dos índices no pacote *TestFraud*, bem como o uso de computadores ou clusters com elevado número de núcleos. Em etapas futuras de pesquisa, serão estudados quais os percentis de desempenho (score ou proficiência) de forma a tornar o processo exequível na prática.

Assim, a próxima seção traz algumas propostas para a melhoria do presente trabalho.

## 6.1 Trabalhos Futuros

Recomenda-se para trabalhos futuros:

- Utilizar processamento paralelo usando rede de computadores.
- Utilizar o pacote *Rcpp* para otimizar a implementação de partes da função responsável pelo cálculo dos índices.
- Estudar de forma mais aprofundada a atribuição de valores para  $E_{min}$  e  $D_{max}$ .
- Analisar valores de  $E_{min}$  e  $D_{max}$  em dados reais.
- Refazer o processo baseado em percentis de desempenho.
- Construir uma estatística  $T$  utilizando ponderação nos índices.
- Utilizar Bases Hierárquicas por área, em que a base na etapa (área do ENEM) 2, utiliza dos pares detectados na etapa 1, e assim em diante.

---

# Referências Bibliográficas

---

- [1] ANDRADE, D. F.; TAVARES, H. R.; VALLE, R. C. *Teoria da Resposta ao Item: conceitos e aplicações*. ABE, São Paulo, 2000.
- [2] BELETI JUNIOR, C. R. *Paralelização de aplicações na plataforma R*. 2013. 95 f. Dissertação (Mestrado em Ciência da Computação) – Departamento de Informática, Universidade Estadual de Maringá, Maringá.
- [3] BOCK, R. D. *Estimating item parameters and latent ability when responses are scored in two or more nominal categories*. *Psychometrika*, 37(1):29–51, 1972.
- [4] CAED - Centro de Políticas Públicas e Avaliação da Educação, 2008. *O que é avaliação educacional?*. Disponível em: <http://www.portalavaliacao.caedufjf.net/pagina-exemplo/o-que-e-avaliacao-educacional/>. Acesso em: 20 dez. 2018.
- [5] CIZEK, G. J.; WOLLACK, J. A. *Handbook of quantitative methods for detecting cheating on tests*. Routledge New York, NY, 2017.
- [6] MOREIRA, F. G. *Cálculo das Proficiências dos Alunos de MG na prova de Matemática do ENEM 2011 através de um modelo da Teoria da Resposta ao Item*. 2014. 82 f. Monografia (Especialização em Matemática) – Instituto de Ciências Exatas, Universidade Federal de Minas Gerais, Belo Horizonte.
- [7] HOLLAND, P. W. *Assessing unusual agreement between the incorrect answers of two examinees using the K-index: Statistical theory and empirical support*. *ETS Research Report Series*, 1996(1):i–41, 1996.
- [8] JONES, M. *Quick Intro to Parallel Computing in R*. Disponível em: <https://nceas.github.io/oss-lessons/parallel-computing-in-r/parallel-computing-in-r.html>. Acesso em: 23 dez. 2018.

- 
- [9] MERSMANN, O. *microbenchmark: Accurate Timing Functions*, 2018. R package version 1.4-6.
- [10] ROCHA, R. *Programação paralela e distribuída: Métricas de Desempenho*. 08-12 de nov. de 2007. 22p. Notas de Aula.
- [11] SOTARIDONA, S. L. *Statistical Methods for the Detection of Answer Copying on Achievement Tests*. 2003. 126 f. Tese de Doutorado apresentada ao Departamento de Metodologia de Pesquisa, Medição e Análise de dados da Universidade de Twente – Holanda.
- [12] SOTARIDONA, L. S.; MEIJER, R. R. *Statistical properties of the K-index for detecting answer copying*. *Journal of Educational Measurement*, 39(2):115–132, 2002.
- [13] SOTARIDONA, L. S.; MEIJER, R. R. *Two new statistics to detect answer copying*. *Journal of Educational Measurement*, 40(1):53–69, 2003.
- [14] VAN DER LINDEN; WIM J.; SOTARIDONA, L. *Detecting answer copying when the regular response process follows a known response model*. *Journal of Educational and Behavioral Statistics*, 31(3):283–304, 2006.
- [15] WESTON, S.; CALAWAY, R. *Getting Started with doParallel and foreach*. 2018.
- [16] WOLLACK, J. A. *A nominal response model approach for detecting answer copying*. *Applied Psychological Measurement*, 21(4):307–320, 1997.
- [17] ZOPLUOGLU, C. *CopyDetect: An R package for computing statistical indices to detect answer copying on multiple-choice examinations*. *Applied psychological measurement*, 37(1):93–95, 2013.
- [18] ZOPLUOGLU, C.; CIZEK, G. J.; WOLLACK, J. A. *Similarity, answer copying, and aberrance: Understanding the status quo*. CIZEK, G. J.; WOLLACK, J. A., “*Handbook of quantitative methods for detecting cheating on tests*,” New York, NY: Routledge, pages 25–46, 2017.