
**SENAI CIMATEC**

**PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM COMPUTACIONAL E TECNOLOGIA INDUSTRIAL**

**Mestrado em Modelagem Computacional e Tecnologia Industrial**

**Dissertação de Mestrado**

# Análise de Agrupamento: O problema da identificação de línguas em textos por meio de *bi-gramas*

Apresentada por: Cleônidas Tavares de Souza Júnior
Orientador: Prof. Dr. Renelson Ribeiro Sampaio

Fevereiro de 2018

Cleônidas Tavares de Souza Júnior

# Análise de Agrupamento: O problema da identificação de línguas em textos por meio de *bi-gramas*

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Modelagem Computacional e Tecnologia Industrial, Curso de Mestrado em Modelagem Computacional e Tecnologia Industrial do SENAI CIMATEC, como requisito parcial para a obtenção do título de **Mestre em Modelagem Computacional e Tecnologia Industrial**.

Área de conhecimento: Interdisciplinar

Orientador: Prof. Dr. Renelson Ribeiro Sampaio
*SENAI CIMATEC*

Salvador
SENAI CIMATEC
2018

Ficha catalográfica elaborada pelo Centro Universitário SENAI CIMATEC

S719a Souza Junior, Cleônidas Tavares de

Análise de agrupamento: o problema da identificação de línguas em textos por meio de bi-gramas / Cleônidas Tavares de Souza Junior. – Salvador, 2018.

101 f. : il. color.

Orientador: Prof. Dr. Renelson Ribeiro Sampaio.

Dissertação (Mestrado em Modelagem Computacional e Tecnologia Industrial) – Programa de Pós-Graduação, Centro Universitário SENAI CIMATEC, Salvador, 2018.
Inclui referências.

1. Mineração de dados. 2. Análise de frequência – Pares de letras. 3. Variação linguística. 4. N-gramas. I. Centro Universitário SENAI CIMATEC. II. Sampaio, Renelson Ribeiro. III. Título.

CDD: 620.00113

NDI - 08

# Nota sobre o estilo do PPGMCTI

Esta dissertação de mestrado foi elaborada considerando as normas de estilo (i.e. estéticas e estruturais) propostas e aprovadas pelo colegiado do Programa de Pós-graduação em Modelagem Computacional e Tecnologia Industrial e estão disponíveis em formato eletrônico (*download* na Página Web http://ead.fieb.org.br/portal_faculdades/dissertacoes-e-teses-mcti.html ou solicitação via e-mail à secretaria do programa) e em formato impresso somente para consulta.

Ressalta-se que o formato proposto considera diversos itens das normas da Associação Brasileira de Normas Técnicas (ABNT), entretanto opta-se, em alguns aspectos, por seguir um estilo próprio elaborado e amadurecido pelos professores do programa de pós-graduação supracitado.

SENAI CIMATEC SISTEMA FIEB
Federação das Indústrias do Estado da Bahia

# CENTRO UNIVERSITÁRIO SENAI CIMATEC

**Mestrado Acadêmico em Modelagem Computacional e Tecnologia Industrial**

A Banca Examinadora, constituída pelos professores abaixo listados, aprova a Defesa de Mestrado, intitulada "ANÁLISE DE AGRUPAMENTOS: O PROBLEMA DA IDENTIFICAÇÃO DE LÍNGUA EM TEXTOS POR MEIO DE BI-GRAMAS," apresentada no dia 22 de fevereiro de 2018, como parte dos requisitos necessários para a obtenção do Título de Mestre em Modelagem Computacional e Tecnologia Industrial.

Orientador: **Prof. Dr. Renelson Ribeiro Sampaio**
SENAI CIMATEC

Coorientador: **Prof. Dr. Hernane Borges de Barros Pereira**
SENAI CIMATEC

Membro Interno **Prof. Dr. Valter de Senna**
SENAI CIMATEC

Membro Externo **Prof. Dr. Marcos Grilo Rosa**
UEFS

Dedico este trabalho a Deus, que construiu as equações que regem o universo.

# Agradecimentos

Agradeço a Deus; à minha esposa Lidia; à minha Família; à FAPESB (T.P. 257/2016); ao grupo de pesquisa *Oficina do Saber*; ao grupo de pesquisa *Fuxicos e Boatos*; aos professores e funcionários do Senai pela paciência que tiveram comigo; pelo apoio nos momentos difíceis; pela luz quando estava perdido; pelo incentivo e pelo amor que todos dedicamos às ciências.

Salvador, Brasil
dia 22 de Fevereiro de 2018

Cleônidas Tavares de Souza Júnior

# Resumo

Algoritmos de aprendizado supervisionado baseados em frequência de letras têm sido usados para identificar as línguas de origem de textos; no entanto, eles são imprecisos quando, por exemplo, tentam distinguir os textos nas línguas norueguesa e dinamarquesa. Os objetivos deste trabalho são: (i) identificar padrões na análise de frequência de pares de letras que possam ser utilizados para agrupar os textos que compartilham uma mesma língua; e (ii) identificar os motivos que levam alguns algoritmos baseados em análise de frequência de letras a serem imprecisos da identificação de algumas línguas. A hipótese inicial é que línguas com uma grande quantidade de palavras em comum e que são variedades/dialetos de uma língua dificilmente são diferenciadas umas das outras por meio da análise de frequência de letras. Para testar essa hipótese, foram desenvolvidos dois algoritmos: (i) um para verificar se a análise de frequência de letras gera resultados suficientes para agrupar os textos de mesma língua em um mesmo agrupamento; e (ii) o outro para verificar a quantidade de palavras compartilhadas por algumas línguas. Os resultados obtidos por meio da análise de agrupamentos revelaram que variedades de uma mesma língua permanecem em um mesmo agrupamento; isso sugere uma proximidade entre elas. Este trabalho contribui (i) para os estudos da linguagem, ao apresentar que variedades de uma mesma língua não podem ser diferenciadas por meio de análise de frequência de pares de letras (com escrita alfabética, com alfabeto latino-europeu); e (ii) para as áreas da computação interessadas em processamento de línguas naturais, com algoritmos que, a partir de um conjunto de textos, identificam e agrupam, graficamente, os textos de mesma variedade linguística ou de mesma língua.

Palavras-chave: Análise de agrupamentos, análise de frequência, variação linguística.

# Abstract

Supervised learning algorithms based on letters frequency have been used to identify languages of texts; however, they are inaccurate when, for example, they try to distinguish texts from the Norwegian and Danish languages. The objectives of this work are: (i) to identify patterns in the frequency analysis of pairs of letters that can be used to group texts that share the same language; and (ii) to identify the reasons that lead some algorithms based on frequency analysis of letters to be inaccurate in the identification of some languages. The initial hypothesis is that languages with a large number of common words and that are varieties / dialects of a language are hardly differentiated from one to another by means of letter frequency analysis. In order to test this hypothesis, two algorithms have been developed: (i) one to verify if the frequency analysis of letters generates enough results to group the texts of the same language into the same grouping; and (ii) the other to check the amount of words shared by some languages. The results obtained through cluster analysis revealed that variations of the same language remain in the same cluster; this suggests a closeness between them. This work contributes (i) to the studies of the language, when presenting that varieties of a same language can not be differentiated by means of analysis of frequency of pairs of letters in alphabetic writing (with Latin-American alphabet); and (ii) for areas of computation interested in natural language processing, with algorithms that, from a set of texts, identify, graphically, group texts of the same linguistic or linguistic variety.

Keywords: Clusters Analysis, Frequency analysis, linguistic variation.

# Sumário

# Lista de Tabelas

# Lista de Quadros

# Lista de Figuras

# Lista de Algoritmos

# Lista de Siglas

UTF - *Unicode Transformation Format*
AGNES - *Aglomerative Nesting*
DIANA - *Divisive Analysis*
FM - *Fowlkes & Mallows*
DB - *Davies-Bouldin*

Capítulo Um

# Introdução

## 1.1 Considerações iniciais

As línguas mudam ao longo do tempo, incorporam palavras de outras línguas e, mesmo com essa incorporação, apresentam características que as distinguem umas das outras. Algoritmos baseados em análises de frequências têm identificado as línguas de origem de diferentes textos, mas apresentam certas imprecisões ao tentar identificar línguas como, por exemplo, o norueguês e o dinamarquês; e o sérvio e o croata. Essas imprecisões contribuem para as discussões a respeito do entendimento do que é uma língua, do que são variedades de uma língua, assim como, de questões relacionadas às políticas linguísticas.

Percebe-se, por meio dos resultados obtidos neste trabalho, que a análise de frequências de letras das línguas apresentam padrões específicos para cada uma delas, exceto para aquelas que são variedades de uma mesma língua. No decorrer deste trabalho, serão apresentados diferentes algoritmos, discussões e resultados que colaboram para o entendimento e identificação das variedades/dialetos das línguas.

## 1.2 Definição do problema

Na área da computação, pesquisas que trabalham com a identificação de línguas em textos por meio de análise de frequência de combinações de letras, chamada *n-gramas*, têm revelado que textos de algumas línguas são mais difíceis de serem distinguídas. Nas áreas dos estudos da linguagem, muitos critérios precisam ser considerados para a identificação e classificação das línguas. Essas duas áreas têm em comum o levantamento de propostas que indiquem quais padrões podem caracterizar cada língua, aproximando-as ou distinguindo-as. Nesse sentido, este trabalho volta-se para as questões:

(1) A análise de *bi-gramas* [1] e a análise de agrupamentos são capazes de reunir conjuntos de textos que partilham uma mesma língua?

(2) Os textos de línguas muito próximas apresentam *bi-gramas* com padrões muito próximos. A análise de agrupamentos consegue distinguir e agrupar os textos considerando as línguas em que foram produzidos?

[1] Análise de frequência de combinações de duas letras.

## 1.3 Objetivo

Os objetivos deste trabalho são: (i) encontrar padrões na análise de frequências de pares letras que possam ser usados para agrupar textos de mesma língua; (ii) identificar os motivos que levam alguns autores à afirmação de que os algoritmos baseados em *n-grama* têm dificuldade em distinguir a língua de textos de línguas muito próximas.

## 1.4 Importância da pesquisa

A pesquisa neste trabalho é de caráter interdisciplinar. É possível dizer que seus resultados contribuem para a pesquisa em mineração de dados e análise de agrupamentos, pois ao considerar certas propriedades das línguas naturais, esclarece por que alguns algoritmos são imprecisos no processamento de línguas naturais. No que diz respeito aos estudos da linguagem, este trabalho apresenta algoritmos que processam grandes quantidades de dados e identificam fenômenos relacionados a variação linguística e ao contato linguístico.

As línguas naturais podem ser observadas e caracterizadas sob diferentes aspectos; podem identificar e caracterizar uma comunidade de maneira política e social; têm características e padrões próprios podendo ser compartilhadas por diferentes comunidades. Os algoritmos propostos neste trabalho agrupam textos de variedades linguísticas de mesma língua inferindo, dessa forma, proximidade entre línguas que, por motivos políticos ou sociais, receberam diferentes nomes.

## 1.5 Motivação

É natural que algumas línguas se extingam e novas línguas surjam ao longo do tempo. Esse fato faz com que as quantidades de línguas vivas, faladas no mundo, sempre mudem. A motivação para o desenvolvimento de algoritmos para o processamento de línguas naturais está em: facilitar o processamento e análise do comportamento dessas línguas contribuindo para a redução de tempo de análise do seu funcionamento; explicitar padrões linguísticos que explicam o funcionamento da linguagem para diferentes grupos de falantes; e contribuir para estudos sobre contatos sociais e históricos entre comunidades.

## 1.6 Limites e limitações

Os limites impostos a este trabalho dizem respeito à coleta, metodologia e análise dos dados e podem ser classificados em: (1) a delimitação do conjunto de línguas e caracteres; (2) a delimitação do conjunto de textos a serem analisados; (3) a escolha da Modelagem utilizada; e (4) a definição dos aspectos metodológicos empregados. De modo detalhado:

(1) a delimitação do conjunto de línguas e caracteres: para este trabalho foi feito um recorte e foram considerados textos em trinta línguas[2] em caracteres latinos/europeus;

(2) a delimitação do conjunto de textos a serem analisados: foram analisados textos escritos, a saber, a Eneida[3], a Bíblia, e alguns conjuntos de textos de outros gêneros (artigos científicos, campanhas publicitárias, sites governamentais, notícias e fóruns de discussão[4]);

(3) a escolha da Modelagem utilizada: os Modelos propostos, por considerarem pares de letras na análise, ainda não consideram aspectos morfológicos, sintáticos ou semânticos;

(4) a definição dos aspectos metodológicos empregados: esta pesquisa analisou as frequências de pares de letras somente por análise de agrupamentos e índices baseados em critérios internos e externos;

## 1.7 Questões e hipóteses

As seguintes hipóteses são discutidas neste trabalho:

(1) a análise de frequência de pares de letras agrupa conjunto de textos de mesma língua; entretanto, não distingue conjuntos de textos de variedades de uma mesma língua;

(2) Muitas palavras em comum entre duas línguas distintas aumentam a imprecisão na identificação e agrupamento de textos nessas línguas;

[2] Ietã, polaca, romena, árabe, sueca, dinamarquesa, norueguesa, francesa, portuguesa, espanhola, latim, italiana, finlandesa, checa, sérvia, croata, húngara, albanesa, alemã, inglesa, indonésia, tagalog, cebuano, maori, haitiana, swahili, chewa, malaia e shuar.

[3] Poema épico escrito por Virgílio no século I a.C. que conta a saga de Eneias ao voltar para casa após o fim da guerra de Troia.

[4] Textos retirados de sites da internet em 2017 (ver Apêndice A)

## 1.8 Aspectos metodológicos

A primeira parte deste trabalho teve como objetivo coletar informações e métodos que pudessem descrever o funcionamento do processo de identificação e classificação das línguas.

Na linguística histórica, a identificação e classificação de línguas podem ser feitas por diferentes métodos. Cada um desses métodos classifica as línguas considerando seus diferentes aspectos morfológicos, sintáticos e/ou semânticos. Alguns desses métodos obtiveram sucesso ao identificar um ancestral comum entre diferentes conjuntos de línguas (e.g., línguas africanas e indoeuropeias).

Na mineração de dados, os algoritmos baseados em aprendizado supervisionado usam análise de frequência de letras para identificar a língua de origem em diferentes textos.

O resultado foi uma pesquisa explicativa que teve como objetivo descobrir porque os algoritmos de aprendizado supervisionado são imprecisos na identificação e agrupamentos de algumas línguas. Para auxiliar a pesquisa, foram desenvolvidos e aplicados dois métodos experimentais: (i) foi criado um algoritmo capaz de processar e agrupar textos com características semelhantes nos arranjos de letras; (ii) foi criado outro algoritmo que cria um diagrama de *Veen* para contabilizar as quantidades de palavras comuns existentes entre as diferentes línguas.

## 1.9 Organização da Dissertação de Mestrado

Para efeito de organização, este trabalho foi organizado conforme exposto abaixo:

- **Capítulo 1 - Introdução**: Contextualiza os objetivos, hipóteses, motivações e problemas da pesquisa; o Capítulo aborda também as limitações, questões relevantes, importância e metodologia adotada.
- **Capítulo 2 - Classificação das línguas**: Descreve, brevemente, diferentes estudos linguísticos sobre a identificação e classificação das línguas. O Capítulo exibe os métodos que Greenberg (2005) e Sapir (2004) desenvolveram para classificar as línguas.
- **Capítulo 3 - Estudos sobre frequências de letras e palavras**: Diferentes algoritmos de análise de frequência de letras/palavras têm mostrado que as características das línguas podem influenciar no modo como esses algoritmos trabalham.

Esse Capítulo descreve alguns resultados encontrados por diferentes autores e que ajudam a esclarecer questões e diretrizes dessa pesquisa.

- **Capítulo 4 - Mineração de dados**: O Capítulo descreve algumas definições relativas à mineração de dados e análise de dados multivariada.
- **Capítulo 5 - Análise de agrupamentos linguísticos**: Apresenta dois Modelos. O primeiro é um algoritmo que usa análise de frequência de pares de letras para agrupar graficamente textos que compartilham a mesma língua. O segundo Modelo usa o diagrama de *Venn* para mostrar a quantidade de palavras em comum entre as línguas. O Capítulo apresenta resultados da pesquisa e parâmetros de configuração da análise de agrupamentos.
- **Capítulo 6 - Considerações finais**:Os resultados sugerem que a análise de frequência de letras é capaz de agrupar as línguas de um texto, mas é imprecisa no agrupamento de variedades de uma mesma língua. Em trabalhos futuros, pretende-se desenvolver um novo algoritmo capaz de calcular as probabilidades de um texto pertencer, ou não, a uma variedade linguística específica.

Capítulo Dois

# Classificação de línguas

A classificação/agrupamento das línguas pode seguir diferentes critérios. As línguas podem ser agrupadas, por exemplo, de acordo com a proximidade geográfica (classificação geográfica); de acordo com a proximidade tipológica (classificação tipológica); ou de acordo com o parentesco (classificação genética).

Nas pesquisas sobre linguagem, existem áreas interessadas nos estudos referentes à classificação das línguas (por exemplo, a construção de tipologias linguísticas). Algumas propostas de classificação estão interessadas em agrupar as línguas com base na similaridade de certas estruturas; outras propostas estão interessadas em identificar um ancestral comum entre elas.

Neste Capítulo, será exposto um resumo de propostas de classificação das línguas. O objetivo é apresentar áreas da linguística que se dedicam a levantar critérios relevantes para analisar, identificar, classificar e agrupar diferentes línguas.

## *2.1 Critérios de classificação*

A classificação geográfica agrupa línguas considerando a proximidade geográfica e assume que o contato entre os falantes de línguas diferentes pode permitir o compartilhamento de certos traços linguísticos e não linguísticos. A classificação geográfica considera que existem regiões onde uma determina língua ocorre e que essa região não depende da existência de fronteiras políticas entre países ou estados. Petter (2015) chama essas regiões de áreas linguísticas e a classificação que segue esse critério de *classificação areal* (geográfica).

As áreas linguísticas podem mudar ao longo dos anos devido aos movimentos populacionais que levam as línguas para diferentes partes do mundo. Isso explicaria a existência de uma mesma língua em regiões distantes umas das outras. A língua portuguesa, por exemplo, é falada em diferentes regiões do mundo como Portugal, Brasil, Cabo Verde, São Thomé e Príncipe, Angola, Moçambique, etc. Esse fato pode ser explicado pela presença portuguesa na América, na África e na Ásia, especialmente, durante o período de exploração colonial.

A classificação tipológica, por sua vez, agrupa línguas que compartilham certas propriedades estruturais (sejam fonológicas, morfológicas, sintáticas ou semânticas). Para ilustrar, o português e o francês, mesmo que historicamente descendam do latim, estariam em gru-

pos distintos se comparados no que diz respeito à possibilidade de realização de sujeito nulo[1] (português) e não realização de sujeito nulo (francês). Esse tipo de classificação não revela ancestrais linguísticos comuns entre as línguas.

A classificação genética, por seu turno, reúne, em um mesmo grupo, línguas que derivam de um ancestral comum. Por exemplo, português e francês podem ser classificadas como línguas neolatinas, da família do indo-europeu. A classificação genética é usada para inferir relações de parentescos entre as línguas. Essas relações podem ser representadas graficamente por meio de estruturas hierárquicas nas quais os níveis mais inferiores representam as línguas e os demais níveis representam as protolínguas (i.e., protolíngua é o nome dado a uma língua hipotética ou reconhecidamente documentada que deu origem a outras línguas; por exemplo, a chamada língua indo-europeia é uma protolíngua hipotética que deu origem a algumas línguas européias e asiáticas). Os critérios para a classificação genética das línguas levam em consideração aspectos geográficos, tipológicos e históricos.

## *2.2 Linguística histórica*

A Linguística Histórica está interessada em analisar as mudanças que aconteceram nas línguas ao longo dos anos assim como as influências de outras línguas na construção de uma língua específica.

As mudanças ocorridas em línguas de tradição oral (i.e., línguas que não registram sua história por meio da escrita) são mais difíceis de serem recuperadas[2] , sendo mais fácil analisar as possíveis mudanças em curso.

As línguas com tradição da escrita podem buscar as mudanças ocorridas em textos antigos comparados aos mais recentes. Uma maneira de analisar as mudanças é por meio do método histórico-comparativo. Lyons (2009) descreve que esse método foi muito usado entre 1820 e 1870 por diversos pesquisadores que, na época, desejavam estabelecer relações entre diferentes línguas comparando as formas e os significados das palavras. Por exemplo, o Quadro 2.1 ilustra uma comparação entre latim, francês, italiano e espanhol.

[1]Sujeito nulo: apagamento do sujeito (i.e., a não realização fonética do sujeito). Por exemplo, em português, as sentenças (1) e (2) são construções sintáticas possíveis: (1) _ *Fui à escola* versus (2) *Eu fui à escola*; em francês a sentença (3), em contraste com a sentença (4), não é uma construção possível: (3)*_ *suis allé à l'école* versus (4) *Je suis allé à l'école.*

[2]Os trabalhos de reconstrução sugerem hipóteses sobre como eram as línguas.

Quadro 2.1: Comparação entre línguas.

| Português | Latim | Francês | Italiano | Espanhol |
|---|---|---|---|---|
| cão | Canis | Chien | Cane | Perro |
| cavalo | Caballus | Cheval | Cavallo | Caballo |

Fonte: Adaptação de Lyons (2009)

Para Lyons (2009), esse tipo de comparação levava os pesquisadores a conclusões simplórias demais e não deixava claro se essas palavras são semanticamente relacionadas. É comum as palavras caírem em desuso e serem substituídas e isso dificultaria a identificação de parentesco entre as expressões. Por exemplo, em espanhol, a palavra latina *canis* acabou sendo substituída por *perro*. A palavra *caballus* era usada para nomear um cavalo específico para carga enquanto a palavra *equus* era usada para fazer referência a um cavalo de maneira geral; o significado inicial de *caballus* se perdeu e assumiu um significado mais genérico.

De acordo com Lyons (2009) não há um consenso entre os pesquisadores do porquê as línguas mudam com o tempo, no entanto, há alguns processos que podem ajudar a entender como as línguas mudam. Segundo Faraco (2005), estudos socioculturais revelam que os falantes mais novos tendem a incluir em seus repertórios mais neologismos do que os falantes mais velhos. Fatores como o contato com outras línguas também podem desencadear processos de mudança.

Em termos gerais, os estudos realizados pelos pesquisadores dos século XIX, quando discutiam os estágios antigos e faziam processos de reconstrução do passado das línguas, agrupavam as línguas segundo a possibilidade de possuírem ancestrais comuns.

Para Sapir (2004), o modo como as comunidades de falantes constroem seus léxicos pode ser usado para classificar as línguas faladas por elas. Por exemplo, um dos métodos usados por Sapir (2004) considera que as formações de palavras podem ser feitas por meio de: (i) conceitos básicos (concretos): são palavras formadas para a indicação de objetos, ações ou qualidades (e.g., a palavra *pé*, grosso modo, significa a parte que sustenta um corpo); (ii) conceitos derivacionais (menos concreto): são as palavras cuja formação tem afixos para acrescentar significância ao radical da palavra (e.g., em *pé-de-couve* a palavra *pé* carrega para a palavra *couve* o significado de *parte que sustenta*); (iii) conceitos relacionais (não puramente abstrato): são palavras cuja formação usam afixos que modificam o seu radical (e.g., *-mente* em *amorosamente* que muda a categoria lexical de *amor*); (iv) conceito relacional puro (abstrato): são palavras cujas formações são originárias de outras palavras (e.g., *pé-de-moleque* ). Com base nesses quatro conceitos Sapir propõe que as línguas podem ser classificadas em:

- Línguas puramente relacionais não derivativas (*Pure-relational non-deriving language*): línguas que mantêm relacionamentos sintáticos puros, sem causar nas palavras grandes modificações por afixos (e.g., formado pelos critérios (i) e (iv)).
- Línguas puramente relacionais complexas (*Complex Pure-relational languages*): línguas que têm relações sintáticas puras e também possuem o poder de modificar os radicais das palavras (e.g., formadas pelos conceitos (i), (ii), (iii) e (iv)).
- Línguas simples de relacionamentos mistos (*Simple Mixed-realtional languages*): línguas que têm um misto de regras entre uma sintaxe rígida (e.g., formados pelos processos (i) e (iii)).
- Línguas complexas de relacionamentos mistos (*Complex Mixed-relation languages*): apresenta características das línguas simples de relacionamentos mistos e considera também as línguas com característica aglomerativa e inflexiva (e.g., formados pelos processos (i), (ii) e (iii)).

## 2.3 Linguística genética

O termo linguística genética é usado para definir os estudos sincrônicos [3] de identificação de traços linguísticos comuns entre as línguas de modo a indicar aquelas que possuem ancestrais comuns.

Greenberg (2005), ao pesquisar tipologia e classificação, coletou e analisou dados de línguas na África, América, Eurásia e Oceania. Em seus estudos sobre a classificação das línguas africanas, Greenberg (2005) formalizou e observou os seguintes princípios:

- Não existe diferença entre o modo como línguas de tradição oral (i.e., sem escrita) e as línguas com tradição da escrita mudam ao longo do tempo, e, por isso, é possível aplicar nesses dois tipos de línguas o mesmo método de classificação.
- É importante excluir do método de classificação das línguas evidências que não sejam linguísticas como, por exemplo, conceitos referentes a raças e crenças.
- O método de classificação das línguas não pode se basear somente nas suas similaridades tipológicas ou semânticas. O método não pode se fundar somente pela presença, ou ausência, de determinados fonemas, classes de palavras ou significados. É importante analisar os dois como um conjunto *forma-significado* (do inglês *Form–meaning*).

[3] Em termos gerais, os estudos sincrônicos consideram os aspectos que as línguas apresentam em um determinado período do tempo. Estudos diacrônicos consideram os estados das línguas através do tempo (SAUSSURE, 1969).

A partir desses princípios, Greenberg (2005) propôs um método que ficou conhecido como Comparação Multilateral. Esse método infere uma classificação com base nas semelhanças entre os pares *forma-significado* das línguas analisadas.

A aplicação desse método requer atenção a alguns detalhes, por exemplo, duas palavras com formas parecidas podem sofrer variação se traduzidas por pessoas diferentes e se as palavras tiverem mais de um significado. Além disso, o autor alerta que o par *forma-significado* pode estabelecer semelhança entre as línguas de quatro formas diferentes (duas de origem histórica e duas de origem não histórica). As semelhanças de origens não histórica podem ocorrer por: (i) acidente, acaso ou coincidência; (ii) simbolismo sonoro, isto é, sons da voz que carregam em si significados (e.g., *mami* é um som que remete à palavra *mãe* em português; *madre* em espanhol; *mother* em inglês, etc). Já as semelhanças de origem histórica podem ocorrer por: (i) empréstimos linguísticos; (ii) uma origem comum entre as línguas.

Com o intuito de reduzir o efeito do acaso na classificação, Greenberg (2005) criou os seguintes passos: (i) calcular a probabilidade de uma combinação de sons ocorrer nas estruturas fonêmicas da língua; (ii) calcular a probabilidade do par *forma- significado* ocorrer aleatoriamente nas línguas que estão sendo comparadas; e (iii) comparar uma amostra de pares de línguas que são reconhecidamente da mesma origem e determinar o quanto são semelhantes entre si. Mesmo assim, as semelhanças feitas por acaso entre as línguas aparecem quatro por cento entre os sistemas fônicos diferentes e sete por cento entre os sistemas fônicos semelhantes. Dessa forma, Greenberg (2005) propõe que semelhanças acima de vinte por cento não sejam consideradas um acaso.

Para calcular e classificar as semelhanças entre as línguas, Greenberg (2005) considera que: (i) a relação *forma-significado* (i.e., representação por fonemas) é menos importante do que a relação *som-significado* (i.e., realização acústica); (ii) quanto maiores forem as semelhanças entre *forma-significado*, maior será o peso atribuído a essa comparação; (iii) as alternâncias alomórficas (i.e., diferentes formas de realização do mesmo morfema ex: *i*moral, *in*feliz, etc.) estabelecem uma conexão histórica entre as línguas; (iv) os processos morfológicos raros também estabelecem conexões históricas (e.g., os infixos); (v) deve-se estabelecer um maior peso para a combinação de morfemas que compartilham *forma-significado* semelhantes entre as línguas; (vi) uma semelhança dada ao acaso entre *forma-significado* diminui exponencialmente à medida que mais línguas vão compartilhando essa propriedade; (vi) se existe semelhança de uma forma-significado entre as línguas A e B, e existe semelhança entre as línguas B e C, então A, B e C recebem maior peso, estabelecendo assim um ancestral (i.e., protolíngua) em comum. Greenberg (2005) considera que um grande número de semelhanças entre esses fatores auxilia o trabalho de classificação das línguas.

## *2.4 Língua e dialeto*

É sabido que as línguas não são blocos homogêneos; ou seja, uma das características das línguas naturais é a possibilidade de variação (cf. Preti (2000)). A existência de variedade de línguas faladas em todas as partes do mundo (variação interlínguas) e a variação encontrada dentro de uma mesma língua (variação intralíngua) ilustra que elas variam no tempo e no espaço. O português, por exemplo, falado no Brasil inclui diferentes formas: o falar da comunidade de falantes do estado do Rio de Janeiro, em alguns aspectos, difere do falar do Rio Grande do Sul. Uma língua é um feixe, um conjunto de variedades, também chamadas de *dialetos*.

O dialeto registra, de alguma forma, uma região, uma cultura, ou uma política compartilhada por uma comunidade de falantes. Petter (2015) argumenta que a distinção entre uma língua e seus dialetos está relacionada a um caráter oficial para a língua e a um caráter não oficial para os dialetos. Muitas línguas oficiais hoje, no passado foram dialetos menos privilegiados de outras línguas chamadas oficiais na época. Desse ponto de vista, a definição de uma língua tem, muitas vezes, mais implicações sociopolíticas do que estruturais. Para Faraco (2005), cada língua tem um conjunto de variedades, e cada variedade caracteriza um grupo de falantes e suas experiências socioculturais e históricas.

Faz-se necessário ressaltar que o presente trabalho propõe uma forma de agrupamento das línguas dos textos em análise e isso é feito por meio de pares de letras.

Capítulo Três

# Estudos sobre frequências de letras e palavras

As letras do alfabeto de uma língua são usadas para representar os sons dessa língua, mas nem sempre elas representam exatamente o som que lhe é atribuído; por exemplo, a letra *r* da palavra *porta* pode ser pronunciada de diferentes formas no português brasileiro: um falante nascido no Rio de Janeiro pronuncia esse *r* de forma diferente de um falante nascido em Salvador. Essas diferenças são chamadas de variações linguísticas.

Como essa diferença de pronúncia não é registrada pelas letras, na escrita não é possível identificar as variações fonéticas entre comunidades de falantes da mesma língua. Por outro lado, as comunidades de falantes tendem a usar expressões pré-fabricadas específicas com maior frequência do que outras comunidades de falantes. Por exemplo, em Portugal, o termo *camisola*, em contextos relacionados ao futebol, é mais usado do que o termo *camisa*. Silva (2008), em um estudo sobre a divergência e convergência entre o português europeu e o português brasileiro com expressões futebolísticas, identificou que existe uma variação mais acentuada entre itens lexicais do que nas estruturas sintáticas.

Bybee (2016) considera que um termo se fortalece no vocabulário dos falantes se ele for usado com frequência. Esse fortalecimento do termo faz com que seja escolhido primeiro pelo processo cognitivo de construção de sentenças do que outro termo menos usado.

Vitral (2006), com o auxílio da análise de frequência de expressões, evidenciou a passagem linguística do emprego do verbo *ter* como lexical (acepção de posse, mais concreta) para gramatical (aspectual[1] ; acepção mais abstrata). Por exemplo, no que diz respeito a acepção de *ter* existencial (acepção mais abstrata), ele aparece em corpus de período contemporâneo e não aparece em corpus dos períodos arcaico e moderno; ou seja, o *ter* com acepção existencial (*De a muitos anos, dia a atrás dia tem a hora de um perdigueiro dormir.*)[2] é mais recente no português. Segundo a análise do autor, o verbo *ter* passou por um processo de gramaticalização[3] apresentando um deslocamento de uma acepção mais concreta (possuir a posse de) para uma acepção mais abstrata (existencial).

Os estudos de frequência e/ou ocorrências de letras/palavras têm sido usados para desenvolver algoritmos de criptografia, compactação de arquivos, identificação de autorias,

[1] De modo geral, Travaglia (2016) explica que o aspecto está relacionado a duração da ação do verbo (por exemplo, aspecto durativo ou pontual).

[2] Exemplo retidado de Vitral (2006)

[3] O processo de gramaticalização implica identificar os processos que levaram um item lexical a se tornar um item gramatical ao longo do tempo (PEZATTI, 2011). Pezatti (2011) apresenta o seguinte exemplo, na sentença *ele tem um carro novo* o verbo *tem* apresenta a função lexical; na sentença *ele tem comprado bugigangas*, *tem* apresenta a função gramatical (verbo auxiliar, aspecto).

identificação de línguas, entre outros.

A análise de frequência de letras e palavras pode também ser usada como uma ferramenta para decodificar alguns tipos de mensagem. Riza *et al.* (2010) sugerem que a análise de frequências de letras e palavras revela um padrão que pode ser usado para decodificar mensagens criptografadas em espanhol. A criptografia é um processo usado para codificar uma mensagem de modo que só o emissor e o receptor possam decodificá-la (i.e., descriptografar) e entendê-la. Segundo os autores, o processo de decodificação pode ser mais trabalhoso dependendo do tipo de texto a ser decodificado. Poesias, por exemplo, tendem a repetir muitas vezes alguns tipos de palavras e letras, o que gera certa imprecisão no processo de decodificação proposto pelos autores.

Moreno (2005) identificou em textos da Bíblia inglesa, dos períodos antigo, moderno e contemporâneo, variação de frequências de letras e palavras. Por exemplo, as palavras do inglês antigo *niht* e *ecg* tornaram-se *night* e *edge*, no inglês contemporâneo. O autor sugere cautela aos programadores interessados em criptografia baseada em análise de frequências de letras e palavras, pois elas variam conforme o período em que os textos foram produzidos e podem gerar erros de decodificação.

Pande & Dhami (2010), através da análise de frequências, identificaram a ocorrência da *Lei de Zipf*[4] em conjuntos de letras e palavras em textos escritos em hindi. Os autores argumentam que, nessa língua, mesmo que os escritores tenham autonomia para escolher e usar diferentes combinações de palavras, os seus textos tendem a seguir algumas regras simples (como a *Lei de Zipf*).

Para que se obtenha resultados conclusivos, a análise de frequência de letras e palavras depende de um *corpus* cujos textos tenham pelo menos uma característica em comum. Hitchcock (1979) verificou que a frequência de certas letras, nas palavras cruzadas, depende da língua usada para a construção do jogo; por exemplo, a letra *e* é mais comum em palavras cruzadas em inglês, francês, italiano e espanhol, mas não em português. No entanto, o autor alerta que a frequência de letras e suas combinações podem variar nos jogos de palavras cruzadas, especialmente, porque são incluídos no jogo abreviações de palavras, siglas, palavras em outras línguas, nomes próprios (e.g., nomes de cidades, países, personalidade, etc).

Em relação aos estudos que usam a análise de frequência como ferramenta para identificação da língua de um texto, Takci & Sogukpinar (2004) sugerem o uso de frequência de letras ao invés de frequência de palavras, pois isso exige menos variáveis de controle.

[4] A Lei de Zipf é uma lei de potências baseadas na distribuição de palavras em um texto. Nessa lei, a frequência de ocorrência de uma palavra em um texto está relacionada à ordem em que ela aparece na classificação das frequências das palavras.

Para os autores, o algoritmo de *Naive Bayes*[5] foi mais preciso na identificação das línguas dos textos analisados. Para a identificação de textos escritos em inglês, francês, alemão e turco o algoritmo registrou acertos entre 97% e 98%.

Ahmed, Cha & Tappert (2004) mostraram que o método de adição cumulativa de frequências de *n-gramas* (i.e., conjuntos de n combinações de letras) gera resultados mais rápidos e assertivos na identificação da língua de textos (e.g., inglês, espanhol, italiano, dinamarquês, sueco, português, alemão, francês, romeno, holandês e tagalog) do que os métodos baseados em algoritmos de *Naive Bayes* e ordenação de *n-grama*.

Ao analisarem mais de 200 línguas, Cazamias, Dixit & Marek (2015) encontraram dificuldades na identificação/distinção entre sérvio, bósnio e croata, de um lado, e indonésio e malaio, de outro.

Como apresentado, os estudos baseados em análise de frequência de letras/palavras mostram que as frequências dentro de uma língua variam ao longo do tempo; evidenciam que existe um padrão de frequência próprio de cada língua, o que permite identificar a língua de origem de textos com o auxílio de algoritmos de aprendizado supervisionado; além disso, os estudos revelam que, para algumas línguas, os algoritmos geram resultados com certa imprecisão.

[5] Naïve Bayes é um algoritmo usado para classificação de dados por meio de cálculos de probabilidades condicionais. Em termos gerais, o algoritmo calcula a probabilidade de um exemplar x pertencer a um agrupamento com base em um conjunto de dados pré-treinados (SILVA; PERES; BOSCARIOLI, 2016).

Capítulo Quatro

# Mineração de dados

A mineração de dados é definida por Tan, Steinbach & Kumar (2009) como um conjunto de técnicas organizadas usadas para descobrir informações úteis em grandes bancos de dados. A mineração de dados tem sido usada para descobrir padrões ou anomalias em grandes quantidades de dados e para auxiliar na tomada de decisão que envolve muitas variáveis.

Na mineração, os dados basicamente passam por três processos: pré-processamento, mineração e pós-processamento. No pré-processamento, os dados brutos são reestruturados de modo a facilitar a coleta e o processamento de informações; nessa reestruturação as seguintes etapas podem ser realizadas: remoção de dados, fusão de dados de diferentes fontes, seleção de dados específicos, etc. No processo de mineração, os dados pré-processados são usados por métodos estatísticos/computacionais para gerar resultados que possam contribuir com os objetivos do trabalho. No pós-processamento, os resultados gerados são visualizados, interpretados e usados para tomadas de decisões e/ou conclusões.

Existem diferentes técnicas e algoritmos que a mineração pode usar. As técnicas de análise de dados multivariada, por exemplo, utilizam diferentes algoritmos para analisar e relacionar diferentes variáveis. Hair *et al.* (2009) classificam as técnicas da análise de dados multivariadas com base em objetivos específicos de uma pesquisa; por exemplo, se na pesquisa existir relações de dependência entre as variáveis, os autores sugerem o uso das seguintes técnicas: análise de correlação, regressão, modelos lineares de probabilidade, modelagem de equações estruturais e/ou análise multivariada de variância; quando as relações entre as variáveis são de interdependência, eles sugerem o uso de: análise fatorial, análise de agrupamentos, análise de correspondência e/ou o escalonamento multidimensional. Todos os Algoritmos que serão apresentados neste Capítulo estão transcritos no Apêndice C em linguagem R[1] (R-CORE, 2017).

## *4.1 Análise de agrupamentos*

Amaral (2016) define que o termo *agrupamento* pressupõe o uso de uma ou mais técnicas não supervisionadas para agrupar instâncias (i.e., objetos) com características comuns e que, posteriormente, serão classificados. Hair *et al.* (2009) definem a análise de agrupamentos como uma técnica de análise usada para identificação de subgrupos significativos

[1] *R* é um programa voltado para análise e desenvolvimento de algoritmos estatísticos/matemáticos; é uma ferramenta gratuita e está disponível em *http://cran.r-project.org*.

dos objetos de estudo. Para Silva, Peres & Boscarioli (2016) a análise de agrupamentos é um processo usado para descobrir relações entre um conjunto de dados com base em suas características.

A análise de agrupamentos é usada para identificar e classificar grupos que compartilham características semelhantes. Em termos gerais, essa análise envolve três etapas: (i) colocar em uma tabela os objetos e características relevantes para a identificação de suas semelhanças ou dissemelhanças; (ii) calcular as distâncias entre esses objetos; (iii) analisar e categorizar os grupos formados.

A tabela a ser construída na etapa (i) deve ter uma coluna para identificar o nome dos objetos a servem analisados e nas demais colunas devem ser inseridas os atributos desses objetos. Os objetos pesquisados devem ser homogêneos, ou seja, eles devem ter atributos[2] que os classifiquem em uma categoria específica. Por exemplo, na Tabela 4.1 os objetos da pesquisa são diferentes cantores, numerados de 1 a 6, e suas características são definidas por dois atributos, no caso, as notas dadas pelos jurados 01 e 02.

Tabela 4.1: Avaliação dos cantores de 01 a 06 pelos jurados 01 e 02.

| **Candidatos** | **Jurado 01** | **Jurado 02** |
|---|---|---|
| Cantor 1 | 1 | 2 |
| Cantor 2 | 6 | 7 |
| Cantor 3 | 5 | 4 |
| Cantor 4 | 7 | 5 |
| Cantor 5 | 9 | 10 |
| Cantor 6 | 9 | 9 |

Fonte: O autor

Depois de concluída a etapa (i) vem a etapa de cálculo de distância. Essas distâncias são usadas para representar numericamente o quão semelhante são os objetos. Os objetos mais semelhantes apresentam distâncias pequenas entre si e os objetos menos semelhantes apresentam distâncias maiores. Na literatura, existem diferentes métodos que podem ser usados para calcular as distâncias entre os objetos, de acordo com Silva, Peres & Boscarioli (2016), o método *euclidiano* é o mais usado.

$$d_{\vec{x_i}\vec{x_j}} = \sqrt{\sum_{k=1}^{m}(x_{ik} - x_{jk})^2} \tag{4.1}$$

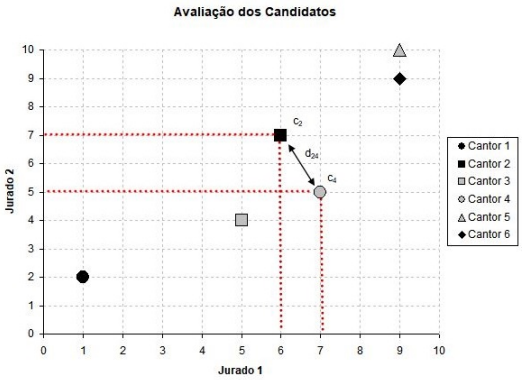
A Equação 4.1 é usada para calcular a distância euclidiana entre dois objetos distintos, $x_i$ e $x_j$. Em relação à nomenclatura de variáveis e conjuntos de dados, será usada a seguinte

[2]Na literatura, os atributos dos objetos também são chamadas de características ou variáveis.

convenção: $\mathbf{X}$ representa um conjunto com $n$ objetos, sendo $\mathbf{X} = \{\vec{x}_1, \vec{x}_2, \vec{x}_3, ...\vec{x}_n\}$. Os objetos $x_i$ e $x_j$ estão contidos em $\mathbf{X}$ e $x_i \neq x_j$. O vetor $\vec{x}_i$ representa todos os atributos do objeto $x_i$, sendo $\vec{x}_i = \{x_{i1}, x_{i2}, x_{i3}...x_{ik}\}$ onde $m$ representa o total de atributos do objeto. Na Equação 4.1 $m$ representa o total de atributos dos objetos e $k$ o valor do atributo para cada objeto.

A Figura 4.1 (a)[3] exibe um gráfico baseado na tabela 4.1, nesse gráfico o eixo $x$ representa uma escala de avaliações para o Jurado 1, o eixo $y$ uma escala de avaliações para o Jurado 2 e os cantores aparecem no gráfico na interseção entre as avaliações dos Jurados 1 e 2. Segue, na Figura 4.1 (b), um exemplo[4] de como calcular a distância euclidiana entre os Cantores 2 e 4.

Figura 4.1: Gráfico e Cálculo da distância euclidiana entre o Cantor 2 e o Cantor 4.




(a) Gráfico.

$$d_{\vec{c_2}\vec{c_4}} = \sqrt{(7-5)^2 + (6-7)^2}$$
$$d_{\vec{c_2}\vec{c_4}} = \sqrt{4+1}$$
$$d_{\vec{c_2}\vec{c_4}} = 2,236$$

(b) Cálculo.

Fonte: O autor

Por meio do método euclidiano, o cálculo da distância entre os Cantores 2 e 4 resulta em 2,236. Depois de calcular as distâncias entre todos os pares de objetos, no caso cantores, é possível identificar o quão semelhantes entre si os objetos são; quanto menor a distância entre dois objetos mais semelhantes são os atributos que eles compartilham.

A Equação 4.1 pode ser generalizada em um algoritmo de modo a automatizar os cálculos.

[3] Essa representação gráfica trabalha com dois atributos; caso o objeto tenha mais de dois atributos, outra forma representação deve ser sugerida.

[4] Nesse exemplo, a variável $x$ foi substituída por $c$, sendo $c_2$ o Cantor 2 e $c_2 = \{6,7\}$ os atributos desse cantor.

**Algoritmo 4.1** Calcular Distâncias Euclidianas

```
1: Método Eq_Euclidiana(V1,V2)
2:     Método Eq_Euclid(v1,v2)
3:         Se Tamanho(v1) = 1 Então
4:             Retorna ((v1[1] − v2[1])^2)
5:         Senão
6:             Retorna Eq_Euclid(v1[−1], v2[−1]) + (v1[1] − v2[1])^2
7:         FimSe
8:     Fim Método
9:     Retorna RaizQuadrada(Eq_Euclid(V1, V2))
10: Fim Método
```

No Algoritmo 4.1, o método *Eq_ Euclidiana(V₁, V₂)* calcula e retorna a distância euclidiana entre dois objetos. Esse método recebe, nos parâmetros de entrada, dois vetores ( $V_1$ e $V_2$); cada um desses vetores contêm os atributos de um objeto específico (e.g., $V_1 = c_2 = \{6,7\}$; $V_2 = c_4 = \{7,5\}$). Na linha 2, o método *Eq_ Euclid(v₁,v₂)* é recursivo e soma iterativamente os quadrados das diferenças entre os atributos de $v_1$ e $v_2$. Na linha 6, o comando [-1] em *v₁[-1]* e *v₂[-1]* representa a retirada do primeiro elemento em cada um desses vetores. Na linha 9, o método *Eq_ Euclid_ Quadrada()* retorna a raiz quadrada do resultado do método *Eq_ Euclid()*.

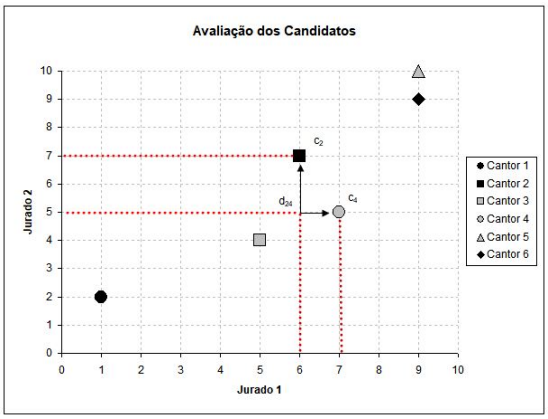
*Manhattan*[5] é o nome de outra métrica usada para calcular as distâncias entre objetos. A Equação 4.2 calcula a distância *Manhattan* entre dois objetos, sendo $m$ o total de atributos do objeto e $k$ o valor do atributo para cada objeto.

$$d_{\vec{x_i}\vec{x_j}} = \sum_{k=1}^{m} \mid x_{ik} - x_{jk} \mid \tag{4.2}$$

No grafico da Figura 4.1 (a), a distância entre os Cantores 2 e 4 é obtida pelo cálculo da hipotenusa a partir dos catetos (diferença entre as avaliações do Jurado 1 e 2). A distância *Manhattan* considera a soma dos catetos. Na Figura 4.2 (a), a linha com setas $d_{24}$ ilustra a distância entre o Cantor 2 e o Cantor 4 (Figura 4.2 (b) exibe o cálculo da distância de *Manhattan* entre os cantores).

[5] Também conhecido como *taxicab, city-block* ou rectilinear.

Figura 4.2: Gráfico e Cálculo da distância *Manhattan* entre o Cantor 2 e o Cantor 4.




(a) Gráfico.

$$d_{\vec{c_2}\vec{c_4}} = |7-5| + |6-7|$$
$$d_{\vec{c_2}\vec{c_4}} = 2+1$$
$$d_{\vec{c_2}\vec{c_4}} = 3$$

(b) Cálculo.

Fonte: O autor

Assim como o método *Eq_Euclid_Quadrada(V₁,V₂)*, o método *Eq_Manhattan(v₁,v₂)*, representado no Algoritmo 4.2, recebe como parâmetro de entrada dois vetores com os atributos dos objetos. Na linha 3, a função *Abs()* retorna o módulo (i.e., um valor positivo) de um número. Na linha 5, existe uma recursividade que passa como parâmetro os vetores $v_1$ e $v_2$ sem suas respectivas primeiras posições.

**Algoritmo 4.2** Calcular Distâncias *Manhattan*

```
1: Método Eq_Manhattan(v1,v2)
2:     Se Tamanho(v1) = 1 Então
3:         Retorna (Abs(v1[1] − v2[1]))
4:     Senão
5:         Retorna Eq_Manhattan(v1[−1], v2[−1]) + abs(v1[1] − v2[1])
6:     FimSe
7: Fim Método
```

Outra métrica é a distância de *Canberra* que é semelhante à distância *Manhattan*. A diferença é que, enquanto a *Manhattan* calcula a diferença absoluta entre as variáveis de dois objetos, a *Canberra* divide essa diferença absoluta entre as variáveis dos dois objetos pela soma dos valores das variáveis absolutas. A Equação 4.3 exibe a distância de *Canberra* para dois objetos.

$$d_{\vec{x}_i\vec{x}_j} = \sum_{k=1}^{m} \frac{\mid x_{ik} - x_{jk} \mid}{\mid x_{ik} \mid + \mid x_{jk} \mid} \tag{4.3}$$

O cálculo da distância *Canberra* entre os Cantores 2 e 4 (i.e., $\vec{x}_i = \vec{c_2} = \{6,7\}$; $\vec{x}_j = \vec{c_4}$

= {7,5}) é: $d_{\vec{c_2}\vec{c_4}}$ = ( |7 - 5|)/(|7|+|5| ) + ( |6 - 7|)/(|6|+|7| ) = 0,243. O Algoritmo 4.3 é uma abstração da Equação 4.3.

**Algoritmo 4.3** Calcular Distâncias Canberra

```
1: Método Eq_Canberra(v1,v2)
2:     Se Tamanho(v1) = 1 Então
3:         Retorna (Abs(v1[1] − v2[1])/(Abs(v1) + Abs(v2))
4:     Senão
5:         Retorna Eq_Canberra(v1[−1], v2[−1]) + Abs(v1[1] − v2[1])/(Abs(v1) + Abs(v2))
6:     FimSe
7: Fim Método
```

O método *Eq_ Canberra(v1,v2)* no Algoritmo 4.3 também recebe como parâmetro de entrada dois vetores de atributos dos objetos.

Existem, na literatura, outros cálculos de distância; alguns são especializados em um tipo de dado, como a distância de *Hamming* que trabalha bem com atributos binários. Esta pesquisa optou por testar esses três cálculos porque são de fácil implantação. Todos os atributos submetidos a esses cálculos serão igual ou maior que zero (atendendo às condições e características dos cálculos) e exigem menos processamento de dados. As três métricas de distâncias (euclidiana, *Manhattan* e *Canberra*) geram, em geral, entre si, resultados diferentes. Essas diferenças, servirão posteriormente, para ajustar os agrupamentos sugeridos aos modelos do mundo real e decidir qual cálculo de distância se adapta melhor a proposta da pesquisa.

Os Algoritmos 4.1, 4.2 e 4.3 calculam a distância entre dois objetos por vez. Para calcular as distâncias entre todas as combinações possíveis de pares de objetos é necessário criar um novo algoritmo.

A Equação 4.4 é usada para calcular o total de combinações de objetos em conjuntos de mesmo tamanho, sendo $n$ o total de objetos e $p$ o tamanho total de cada conjunto.

$$C_{n,p} = \frac{n!}{(n-p)!p!} \tag{4.4}$$

A Tabela 4.1 tem seis objetos (i.e., cantores); para descobrir as combinações possíveis de pares objetos da Tabela 4.1, basta calcular ($n = 6$ (cantores) e $p = 2$ (pares de cantor)) $C_{6,2} = 6!/((6-2)! * 2!) = 15$. A Equação 4.4 aparece na linha 3 do Algoritmo 4.4; o total de combinações possíveis servirá para a construção de uma tabela com todas combinações de pares de objetos e suas respectivas distâncias.

**Algoritmo 4.4** Calcular Distâncias entre objetos de uma tabela

```
1:  Método CALCULO_DE_DISTANCIAS(tabela,metodo)
2:      tot_lin ← TotalDeLinhas(tabela)
3:      tot_comb ← Fatorial(tot_lin)/(Fatorial(tot_lin − 2) ∗ Fatorial(2))
4:      tab_dist ← Matriz[tot_comb][3]
5:      pos ← d ← 0
6:      Para a ← 1 até (tot_lin-1) Faça
7:          Para b ← a+1 até (tot_lin) Faça
8:              v1 ← tabela[a][2 : TotalDeColunas(tabela)]
9:              v2 ← tabela[b][2 : TotalDeColunas(tabela)]
10:             Se ( metodo = 1 ) Então
11:                 d ← Eq_Euclid_Quadrada(v1, v2)
12:             FimSe
13:             Se ( metodo = 2 ) Então
14:                 d ← Eq_Manhattan(v1, v2)
15:             FimSe
16:             Se ( metodo = 3 ) Então
17:                 d ← Eq_Canberra(v1, v2)
18:             FimSe
19:             pos ← pos + 1
20:             tab_dist[pos][1] ← a
21:             tab_dist[pos][2] ← b
22:             tab_dist[pos][3] ← d
23:         FimPara
24:     FimPara
25:     tab_dist ← Classificar(tab_dist[][3], crescente)
26:     Retorna tab_dist
27: Fim Método
```

O Algoritmo 4.4 usa o cálculo de distância *euclidiana*, *Manhattan* ou *Canberra* para calcular as distâncias de todos os pares de objetos de uma tabela. O método *Calculo_De_Distancias(tabela,metodo)* recebe, pelos parâmetro de entrada, uma tabela [6] (e.g., Tabela 4.1) e um número para identificar a métrica de distância a ser usada (e.g., 1 - para usar a distância *euclidiana*; 2 - para a distância *Manhattan*; 3 - para a distância *Canberra*). Como parâmetro de saída, o Algoritmo 4.4 retorna uma tabela com as distâncias (e.g., Tabela 4.2) entre cada par de objeto e classificada da menor para a maior distância. Na linha 4 do Algoritmo 4.4, é criada uma matriz; cada linha dessa matriz guardará um par de objetos e a distância entre esses objetos (e.g., 2;4;2,236, ou seja, Cantor 2, Cantor 4 e a distância de 2,236 entre os cantores). Os objetos serão identificados por números, ou seja, o objeto da primeira linha da tabela carregada é o número 1, o segundo objeto, o

[6] A primeira coluna dessa tabela deve conter objetos e as demais conter os atributos numéricos desses objetos.

número 2 e assim sucessivamente. Nas linhas 8 e 9, $v_1$ e $v_2$ recebem da tabela carregada somente os atributos dos objetos, ignorando a primeira coluna (i.e., o objeto). Na linha 26, o método *Classificar()* ordena todos os pares de objetos da tabela *tab_dist* pela terceira coluna (distância) da menor para a maior; essa classificação facilitará a identificação da menor distância além de colaborar com os processamentos de formação de grupos que serão vistos posteriormente. O método *Calculo_De_Distancias(tabela,metodo)* carregado com *metodo = 1* e *tabela = Tabela 4.1* retorna uma tabela similar a Tabela 4.2.

Tabela 4.2: Tabela de distâncias euclidianas dos objetos da Tabela 4.1.

| **Objeto 01** | **Objeto 02** | **Distâncias** |
|---|---|---|
| 5 | 6 | 1.000000 |
| 2 | 4 | 2.236068 |
| 3 | 4 | 2.236068 |
| 2 | 3 | 3.162278 |
| 2 | 6 | 3.605551 |
| 2 | 5 | 4.242641 |
| 1 | 3 | 4.472136 |
| 4 | 6 | 4.472136 |
| 4 | 5 | 5.385165 |
| 3 | 6 | 6.403124 |
| 1 | 4 | 6.708204 |
| 1 | 2 | 7.071068 |
| 3 | 5 | 7.211103 |
| 1 | 6 | 10.630146 |
| 1 | 5 | 11.313708 |

Fonte: O autor

Como dito anteriormente, na análise de agrupamentos, a semelhança entre dois objetos pode ser identificada pela distância entre eles, quanto menor essa distância mais semelhantes são. Na Tabela 4.2 os objetos mais próximos/semelhantes são 5 e 6 (i.e., os Cantores 5 e 6) e os mais distantes e menos semelhantes entre si são os objetos 1 e 5 (Cantores 1 e 5).

O Algoritmo 4.5 foi construído para facilitar a identificação da distância de pares de objetos que estejam na Tabela 4.2. O Algoritmo 4.5 será útil posteriormente na identificação e consolidação de grupos.

**Algoritmo 4.5** Encontrar distância entre pares de objetos

```
1: Método Distancia_Entre_Dois_Objetos(Obj1, Obj2, tab_dist )
2:     Para i ← 1 até TotalDeLinhas(tab_dist) Faça
3:         Se (Obj1=tab_dist[i][1] E Obj2=tab_dist[i][2]) Ou (Obj1=tab_dist[i][2] E
   Obj2=tab_dist[i][1]) Então
4:             Retorna tab_dist[i][3]
5:         FimSe
6:     FimPara
7: Fim Método
```

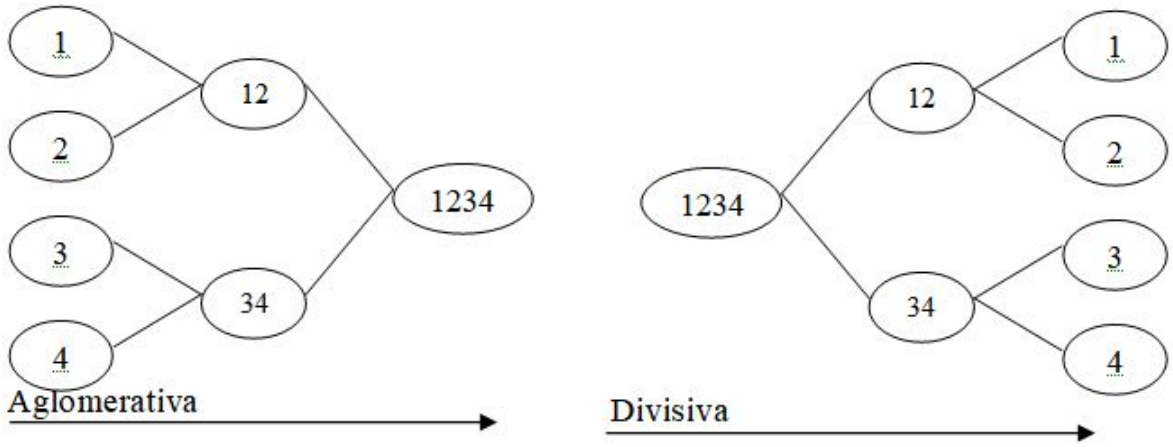
O método *Distancia_Entre_Dois_Objetos(Obj1,Obj2,tab_dist)* no Algoritmo 4.5 recebe como parâmetro de entrada *Obj1*, *Obj2* e *tab_dist*. *Obj1* é uma variável numérica com o valor do primeiro par de objetos a buscar; o *Obj2* recebe o valor do segundo par; e *tab_dist* é a tabela com as distâncias entre os pares de objetos (e.g., Tabela 4.2). O método *TotalDeLinhas()*.

Em relação à formação de grupos, convencionou-se, neste trabalho, que **G** representa um conjunto com $n$ grupos, sendo $\mathbf{G} = \{g_1,g_2,g_3,...,g_n\}$. A representação de grupos $g_i$ e $g_j$ pertence a **G** e $g_i \neq g_j$. O vetor $\vec{g_i}$ representa todos objetos do grupo $g_i$, podendo $\vec{g_i}$ conter, por exemplo, dois objetos (e.g., $g_1 = \vec{g_1} = \{x_2, x_4\}$). Todos os objetos de **X** estão em algum grupo de **G**. Por exemplo, sendo $\mathbf{G} = \{g_1,g_2\}$, $\mathbf{X} = \{x_1,x_2,x_3,x_4,x_5,x_6\}$, $g_1 = \vec{g}_1 = \{x_2, x_5, x_6\}$, $g_2 = \vec{g}_2 = \{x_1, x_3, x_4\}$), então $\mathbf{G} = \{\{x_2, x_5, x_6\}, \{x_1, x_3, x_4\}\}$. Em **G**, cada $x$ deve pertencer a um grupo; não há $x$ em **G** que apareça em mais de um grupo.

Há diferentes maneiras de representar graficamente os agrupamentos de objetos; uma delas é por meio de uma representação hierárquica. Há duas maneiras comuns de se realizar o procedimento hierárquico: (i) Aglomerativo - os objetos mais semelhantes são agrupados primeiro, depois, iterativamente, os sub-grupos mais semelhantes até chegar ao topo da hierarquia; (ii) Divisivo - o agrupamento no topo da hierarquia é iterativamente dividido até se chegar aos objetos estudados.

As estratégias de formação de agrupamentos hierárquico aglomerativos, procedimento (i), são chamadas AGNES (do inglês **A** **G**lomerative **NES**ting; Figura 4.3 a). As estratégias de formação de agrupamentos hierárquico divisivos, procedimento (ii), são chamadas DIANA (do inglês **DI**visive **ANA**lysis; Figura 4.3 b)).

Figura 4.3: Método DIANA e AGNES.

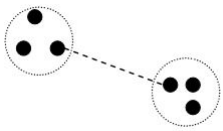
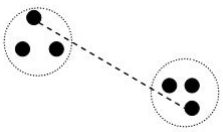
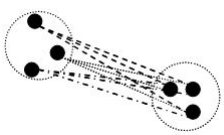



(a) Método AGNES. (b) Método DIANA.

Fonte: Adaptado de Silva, Peres & Boscarioli (2016)

A formação de um novo agrupamento depende de quão próximos estão os grupos uns dos outros. O cálculo de distância entre grupos pode ser feito por diferente métodos. O Quadro 4.1 ilustra alguns deles.

Quadro 4.1: Identificação das distâncias entre Grupos.

| **Representação** | **Descrição** |
|---|---|
|  | *Single Linkage* ou *menor distância*: Para a inclusão de um novo objeto a um grupo, ou a consolidação de dois grupos em um grupo maior, esse método considera a menor distância entre esse novo objeto e o objeto do grupo mais próximo dele, ou a menor distância entre objetos do primeiro grupo e objetos do segundo grupo. |
|  | *Complete Linkage* ou *maior distância*: Esse método considera a maior distância entre um novo objeto e os objetos do grupo que vai incluí-lo, ou a maior distância entre os objetos do primeiro grupo com os objetos do segundo grupo. |
|  | *Average Linkage* ou *distância média*: Esse método calcula a média das distâncias entre um novo objeto e todos os outros objetos do grupo que vai incluí-lo, ou a média de distâncias entre os objetos do primeiro grupo com os objetos do segundo grupo. |

Fonte: Adaptado de Silva, Peres & Boscarioli (2016).

Cada um desses métodos tem vantagens e desvantagem. O método de *menor distância*, Equação 4.5, tem a vantagem de conseguir detectar grupos com formatos não elípticos (e.g., com formato côncavo ou convexo - ver Figura 4.6). Esse método apresenta bons resultados quando associado ao cálculo de distâncias euclidianas, mas é sensível a ruídos [7], tende a formar encadeamentos[8] e não distingue grupos muito próximos.

$$d_{(g_i,g_j)} = min(d_{(\vec{g}_i,\vec{g}_j)}) \tag{4.5}$$

O método de *maior distância*, Equação 4.6, tem a desvantagem de fragmentar grupos muito grandes e a vantagem de ser menos sensível a ruídos, tende a formar grupos compactos, apresenta bons resultados associado ao cálculo de distância euclidiana e favorece a manipulação de grupos com formatos de globo (i.e., círculos ou esperas).

$$d_{(g_i,g_j)} = max(d_{(\vec{g}_i,\vec{g}_j)}) \tag{4.6}$$

O método de *distância média*, Equação 4.7, é mais indicado para formações de grupos em formato de globo; é menos sensível a ruídos do que os métodos de maior e menor distância. Na Equação 4.7, $n_i$ e $n_j$ são os números objetos dos grupos $g_i$ e $g_j$, respectivamente.

$$d_{(g_i,g_j)} = \sum_{ij} \frac{d_{(\vec{g}_i,\vec{g}_j)}}{(\mid n_i \mid + \mid n_j \mid)} \tag{4.7}$$

Tanto grupos como objetos podem ser identificados e armazenados em um mesmo vetor. Com essa abordagem, as posições de um vetor passam a representar os objetos e os valores atribuídos a cada vetor passam a representar o grupo no qual o objeto está inserido. Por exemplo, em $v = \{1,1,2,2,1,2\}$ o vetor $v$ tem dois grupos, 1 e 2, o grupo 1 é composto pelos objetos 1,2 e 5, enquanto o grupo 2 é composto pelos objetos 3,4 e 6. Esse tipo de representação é econômica porque trabalha com duas informações diferentes em uma mesma estrutura e facilita a identificação dos grupos e seus respectivos objetos sem recorrer a outros vetores/estruturas. Essa representação será útil para os algoritmos que calcularão as distâncias entre grupos.

O Algoritmo 4.6 foi desenvolvido para identificar no vetor que guarda objetos e grupos quais são os grupos desse vetor.

[7] São valores considerados diferentes da maioria dos outros valores no mesmo conjunto; podem ser causados por erro no seu registro ou por mudanças repentinas no fenômeno observado.

[8] Incorpora a cada iteração um novo objeto gerando uma longa sequência de objetos e dificultando a identificação de grupos por meio de cortes.

**Algoritmo 4.6** Listar os grupos identificados em um vetor

```
1:  Método ListarGrupos(vetor)
2:      v_resp ← Vetor(Comprimento = 1)
3:      v_resp ← vetor[1]
4:      Para i ← 1 até Tamanho(vetor) Faça
5:          ctrl ← 0
6:          Para j ← 1 até Tamanho(v_resp) Faça
7:              Se (vetor[i] = v_resp[j]) Então
8:                  ctrl ← 1
9:                  Break;
10:             FimSe
11:         FimPara
12:         Se (ctrl = 0) Então
13:             v_resp ← Concatenar(v_resp, vetor[i])
14:         FimSe
15:     FimPara
16:     Retorna v_resp
17: Fim Método
```

O método *ListarGrupos()* no Algoritmo 4.6 recebe como parâmetro de entrada um vetor com objetos e grupos e retorna um vetor com os grupos identificados. Na linha 9 do Algoritmo 4.6 o comando *Break* encerra o *looping* do laço de *j*. Na linha 13, o comando *Concatenar* incluí uma nova posição no vetor *v_resp* com o valor do grupo recém localizado. Por exemplo, sendo $v = \{1,2,2,1,3,2\}$ a atribuição $g \leftarrow$ *ListarGrupos(v)* resultaria em $g = \{1,2,3\}$. Essa identificação dos grupos no vetor é necessária para o cálculo e formação dos grupos nos métodos *menor distância*, *maior distância* e *distância média*. Esses métodos dependem da identificação dos grupos e seus respectivos objetos para fazer as comparações entre grupos e determinar a menor distância entre eles. O Algoritmo 4.7 irá combinar cada grupo identificado no Algoritmo 4.6 e retornar uma lista com todos os pares de grupos possíveis.

**Algoritmo 4.7** Listar os pares de combinações possíveis entre os grupos no vetor

```
1:  Método ListarParesDeGrupos(grupos)
2:      n ← Tamanho(grupos)
3:      n_pares ← Fatorial(n)/(Fatorial(n − 2) ∗ 2)
4:      m ← Matriz[n_pares][2]
5:      tmp ← 0
6:      Para i ← 1 até (n-1) Faça
7:          Para j ← (i+1) até n Faça
8:              tmp ← tmp + 1
9:              m[tmp][1] ← grupos[i]
10:             m[tmp][2] ← grupos[j]
11:         FimPara
12:     FimPara
13:     Retorna m
14: Fim Método
```

O Método *ListarParesDeGrupos()* recebe como parâmetro um vetor com os grupos a serem combinados. Por exemplo, sendo o vetor *grupos*, na linha 1, *grupos = {1,2,3}*, a atribuição *paresdegrupos ← ListarParesDeGrupos(grupos)* retornaria *paresdegrupos= {{1,2},{1,3}, {2,3}}*. Os resultados em *paresdegrupos* (i.e., as combinações possíveis de grupos) serão importantes no processo de identificação e geração de grupos.

O método *ProximoGrupo()*, no Algoritmo 4.8, recebe como parâmetros de entrada uma tabela de distância entre objetos (e.g., Tabela 4.2); uma lista com as combinações de pares possíveis de grupos (e.g., variável *paresdegrupos*); um vetor com objetos e grupos (e.g., *v = {1,2,2,1,3,2}*); e um numero de 1 a 3 (i.e., 1 - para que a identificação de grupos considere a *menor distância* (e.g., Quadro 4.1), 2 - para a *maior distância* ou 3 - para a *distância média*). Como parâmetro de saída, o método *ProximoGrupo()* retornará um vetor com três elementos; os dois primeiros são os pares de grupos/objetos mais próximos e o terceiro é a distância que existe entre esse novo grupo e seus respectivos grupos anteriores. Por exemplo, dado que uma combinação possível de agrupamento para os dados da Tabela 4.1 seja *vetor = {3,2,2,2,1,1}* (i.e., **G** = {$g_1$,$g_2$,$g_3$} onde $g_1$ = {5,6}, $g_2$ = {2,3,4} e $g_3$ = {1}) e, dado que se pretende usar o método *maior distância* para calcular as distâncias intergrupos, o método *ProximoGrupo(tab_dist, paresdegrupos, vetor, 2)* retornaria {3;2;7,071}) sugerindo que os grupos $g_2$ e $g_3$ são os grupos mais parecidos em G e que, depois de agrupar $g_2$ e $g_3$ em um novo grupo, esse novo grupo estará a 7,071 de distância de $g_2$ e $g_3$.

**Algoritmo 4.8** Retornar a próxima formação de grupo

```
1: Método ProximoGrupo(tab_dist,g_pares, vetor, metodo)
2:     n ← TotalDeLinhas(g_pares)
3:     tv ← Tamanho(v)
4:     m ← Matriz[n][3])
5:     cont ← 0
6:     Enquanto n>0 Faça
7:         m[n][1] ← g_pares[n][1]
8:         m[n][2] ← g_pares[n][2]
9:         c_med ← v_med ← 0
10:        d1 ← d2 ← 0
11:        Para i ← 1 até tv Faça
12:            Se ( v[i] = m[n][1] ) Então
13:                Para j ← 1 até tv Faça
14:                    Se (v[j] = m[n][2]) E (i!=j) Então
15:                        d1 ← Distancia_Entre_Dois_Objetos(i, j, tab_dist)
16:                        Se ( d2 = 0 ) Então
17:                            {d2 ← d1}
18:                        FimSe
19:                        Se (metodo = 1) E (d1 < d2) ) Então
20:                            {d2 ← d1}
21:                        FimSe
22:                        Se (metodo = 2) E (d1 > d2) Então
23:                            {d2 ← d1}
24:                        FimSe
25:                        c_med ← c_med + 1
26:                        v_med ← v_med + d1
27:                    FimSe
28:                FimPara
29:            FimSe
30:        FimPara
31:        Se (metodo = 3) Então
32:            m[n][3] ← (v_med/c_med)
33:        Senão
34:            m[n][3] ← d2
35:        FimSe
36:        n = n − 1
37:    FimEnquanto
38:    m ← Classificar(m, crescente)
39:    Retorna Vetor(m[1][])
40: Fim Método
```

O laço *Enquanto*, na linha 6 do Algoritmo 4.8, lê todos os pares de grupos carregados em *g_pares* e, a partir da linha 11, o laço *Para* compara, entre os pares de grupos, as distâncias entre os objetos do primeiro grupo e do segundo grupo 02. A depender do cálculo de distância entre grupos (e.g., *menor distância*, linha 19; *maior distância, linha 22; ou distância média, linha 32*) o algoritmo armazenará as distâncias entre cada par de grupo na matriz *m* cujas duas primeiras colunas armazenam os grupos e a terceira, a distância. Nas linhas 38-39, o algoritmo identifica a menor distância entre grupos e retorna em um vetor os dois grupos eleitos e suas distâncias em relação à formação inicial de grupos (e.g., sugerida na variável *vetor*).

Para facilitar o processamento e a identificação de grupos com um objeto, convencionou-se, neste trabalho, que o vetor que armazena grupos e objetos (i.e., *vetor*={3,2,2,2,1,1}) identificará grupos unitários com valores negativos e grupos com mais de um objeto com valores positivos. Por exemplo, em *vetor* = {-1,-2-,-3,-4,-5,-6} todos os objetos da variável *vetor* estão em grupo próprio (i.e., $\mathbf{G}$={$g_1$,$g_2$,$g_3$,$g_4$,$g_5$,$g_6$}), e, em *vetor* = {-1,-2,-3,-4,1,1}, os grupos são $\mathbf{G}$ = {$g_1$,$g_2$,$g_3$,$g_4$,$g_5$}, sendo $g_1 = \{x_1\}$, $g_2 = \{x_2\}$, $g_3 = \{x_3\}$, $g_4 = \{x_4\}$ e $g_5 = \{x_5, x_6\}$. Essa distinção entre grupos unitários e grupos com mais de um objeto será útil para a identificação de objetos muito distintos. Como a identificação de semelhança entre objetos começa com os mais similares, os menos similares acabam sendo os últimos a entrar em algum grupo e esse tipo de representação (com valores negativos) facilita a identificação desses objetos. Com base nesse critério, a primeira formação de grupos de objetos da Tabela 4.1 é *vetor*={-1,-2,-3,-4,-5,-6}. Ao executar o método *ProximoGrupo(tab_dist,ListarParesDeGrupos(ListarGrupos(v)), vetor, 2)*, obtêm-se {-5;-6;1,0}, ou seja, os objetos 5 e 6 são os mais semelhantes e podem formar o primeiro grupo identificado. Assim *vetor* passa a ser *vetor*={-1,-2,-3,-4,1,1} e ao submeter novamente *vetor* ao método *ProximoGrupo(tab_dist,ListarParesDeGrupos(ListarGrupos(v)), vetor, 2)*, obtêm-se {-2;-4;2,236}, ou seja, a próxima formação de grupos envolverá os objetos 2 e 4 e formará o segundo grupo, *vetor*={-1,2,-3,2,1,1}. Nesse processo de formação de grupos, quando um grupo unitário encontrar um grupo com mais de um objeto, ele receberá o número de identificação do grupo que não é unitário; quando um grupo não unitário encontra outro não unitário, os grupos receberão o número de identificação de menor valor entre eles. Os exemplos de iteração e de formação dos grupos serão visualizados no retorno do Algoritmo 4.9.

**Algoritmo 4.9** Retornar as distâncias entre todos os grupos

```
1: Método GerarGrupos(tabela, metodo_intragrupo, metodo_intergrupos)
2:     tab_dist ← Calculo_De_Distancias(tabela, metodo_intragrupo)
3:     tot_obj ← TotalDeLinhas(tabela)
4:     tot_lin ← TotalDeLinhas(tab_dist)
5:     G ← Vetor[tot_obj + 1]
6:     tab_grup ← Matriz[][]
7:     a ← b ← ct ← 0
8:     Enquanto (b < tot_lin) Faça
9:         extr ← 0
10:        a ← TotalDeLinha(tab_grup)
11:        b ← b + 1
12:        G ← tab_grup[a][]
13:        obj1 ← tab_dist[b, 1]
14:        obj2 ← tab_dist[b, 2]
15:        dist ← tab_dist[b, 3]
16:        Se (tab_grup[a][obj1] < 0) E (tab_grup[a][obj2] < 0) Então
17:            ct ← ct + 1
18:            G[obj1] ← G[obj2] ← ct
19:            G[tot_obj + 1] ← dist
20:        Senão
21:            g ← ListarGrupos(G[De{1}Ate{(Tamanho(G) − 1)}])
22:            Se (Tamanho(g) = 1) Então
23:                Break;
24:            FimSe
25:            g_pares ← ListaParesDeGrupos(g)
26:            md ← ProximoGrupo(tab_dist, g_pares, G[1 : (Tamanho(G) −
    1)], metodo_intergrupos)
27:            Se (md[1]<0) E (md[2]<0) Então
28:                extr ← 1
29:            FimSe
30:            Se (md[1]<0) Ou (md[2]<0) Então
31:                Se (md[1]<0) Então
32:                    z ← md[2]; w ← md[1]
33:                Senão
34:                    z ← md[1]; w ← md[2]
35:                FimSe
36:                Para i ← 1 até (Tamanho(G)-1) Faça
37:                    Se ( G[i] = w ) Então
38:                        G[i] ← z
39:                    FimSe
40:                FimPara
41:                G[Tamanho(G)] ← md[3]
```

```
42:             Senão
43:                 Se (md[1]<md[2]) Então
44:                     z < −md[1]; w < −md[2]
45:                 Senão
46:                     z < −md[2]; w < −md[1]
47:                 FimSe
48:                 Para i ← 1 até Tamanho(G)-1 Faça
49:                     Se (G[i] = w) Então
50:                         G[i] < −z
51:                     FimSe
52:                 FimPara
53:                 G[Tamanho(G)] ← md[3]
54:             FimSe
55:         FimSe
56:         Se (extr = 0) Então
57:             tab_grup ← Concatenar(tab_grup, G)
58:         FimSe
59:     FimEnquanto
60:     Retorna tab_grup
61: Fim Método
```

O Algoritmo 4.9 recebe como parâmetro de entrada uma tabela de objetos e atributos (e.g., Tabela 4.1); um valor numérico de 1 a 3 em *metodo_intragrupo* (1-para medir as distâncias euclidianas entre os objetos da tabela a ser carregada; 2 - para medir com as distâncias Manhattan; e 3 - para medir com a distâncias Canberra.); e um valor numérico de 1 a 3 em *metodo_intergrupos* (1 - para medir a *menor distância* entre os grupos; 2 - para a *maior distância*; 3 - para as *distâncias médias*). O método *GerarGrupos(tabela,1,2)*, no Algoritmo 4.9, calcula as distâncias entre os objetos (linha 2), identifica as semelhanças entre objetos e grupos (linha 16) e retorna uma tabela (e.g., Tabela 4.3 ) na qual cada linha representa uma formação de grupo (a primeira linha apresenta um grupo para cada objeto e a última linha termina com um grupo com todos os objetos). A última coluna dessa tabela armazena a distância do grupo em relação ao grupo anterior e as demais colunas representam cada objeto identificado.

Tabela 4.3: Formação iterativa de grupos

| [Obj1] | [Obj2] | [Obj3] | [Obj4] | [Obj5] | [Obj6] | [Distâncias] |
|---|---|---|---|---|---|---|
| -1 | -2 | -3 | -4 | -5 | -6 | 0.000000 |
| -1 | -2 | -3 | -4 | 1 | 1 | 1.000000 |
| -1 | 2 | -3 | 2 | 1 | 1 | 2.236068 |
| -1 | 2 | 2 | 2 | 1 | 1 | 3.162278 |
| 2 | 2 | 2 | 2 | 1 | 1 | 7.071068 |
| 1 | 1 | 1 | 1 | 1 | 1 | 11.313708 |

Fonte: O autor.

O método *AjustarLayout()* no Algoritmo 4.10 recebe como parâmetro de entrada a tabela de grupos do Algoritmo 4.9 e retorna uma tabela com um *Layout* mais detalhado.

**Algoritmo 4.10** Ajustar o Layout do Resultado

```
1:  Método AjustarLayout(tab_grup)
2:      nlin ← TotalDeLinhas(tab_grup)
3:      ncol ← TotalDeColunas(tab_grup)
4:      df ← Matriz[][]
5:      tmp < −0
6:      Para i ← 1 até nlin Faça
7:          v ← tab_grup[i][De{1}Ate{(ncol − 1)}]
8:          g ← ListaGrupos(v)
9:          p ← Concatenar("{")
10:         Para j ← 1 até Tamanho(g) Faça
11:             p ← Concatenar("{")
12:             Para k ← 1 até Tamanho(v) Faça
13:                 Se g[j]=v[k] Então
14:                     p ← Concatenar(k)
15:                     p ← Concatenar(",")
16:                 FimSe
17:             FimPara
18:             p ← Substituir(p, ultimaposicao, "},")
19:         FimPara
20:         p ← Substituir(p, ultimaposicao, "}")
21:         lin ← Concatenar(c(Tamnaho(g), p, tab_grup[i], [ncol]))
22:         Se tmp = 0 Então
23:             df < −Concatenar(df, lin)
24:             tmp < −1
```

```
        Senão
            NomeDasColunas(df) <- ("NroDeGrupos","Grupos","Distancia")
        FimSe
    FimPara
    Retorna df
Fim Método
```

Esse novo *layout* exibe, na primeira coluna, quantos grupos foram identificados; na segunda coluna, exibe, entre colchetes, os objetos de cada grupo; e, na terceira coluna, exibe as distâncias entre a formação de grupos anterior e a nova formação.

Tabela 4.4: Formação iterativa de grupos - Novo Layout

| NroDeGrupos | Grupos | Distancia |
|---|---|---|
| 6 | {1},{2},{3},{4},{5},{6} | 0 |
| 5 | {1},{2},{3},{4},{5,6} | 1.0000 |
| 4 | {1},{2,4},{3},{5,6} | 2.2360 |
| 3 | {1},{2,3,4},{5,6} | 3.1622 |
| 2 | {1,2,3,4},{5,6} | 7.0710 |
| 1 | {1,2,3,4,5,6} | 11.3137 |

Fonte: O autor.

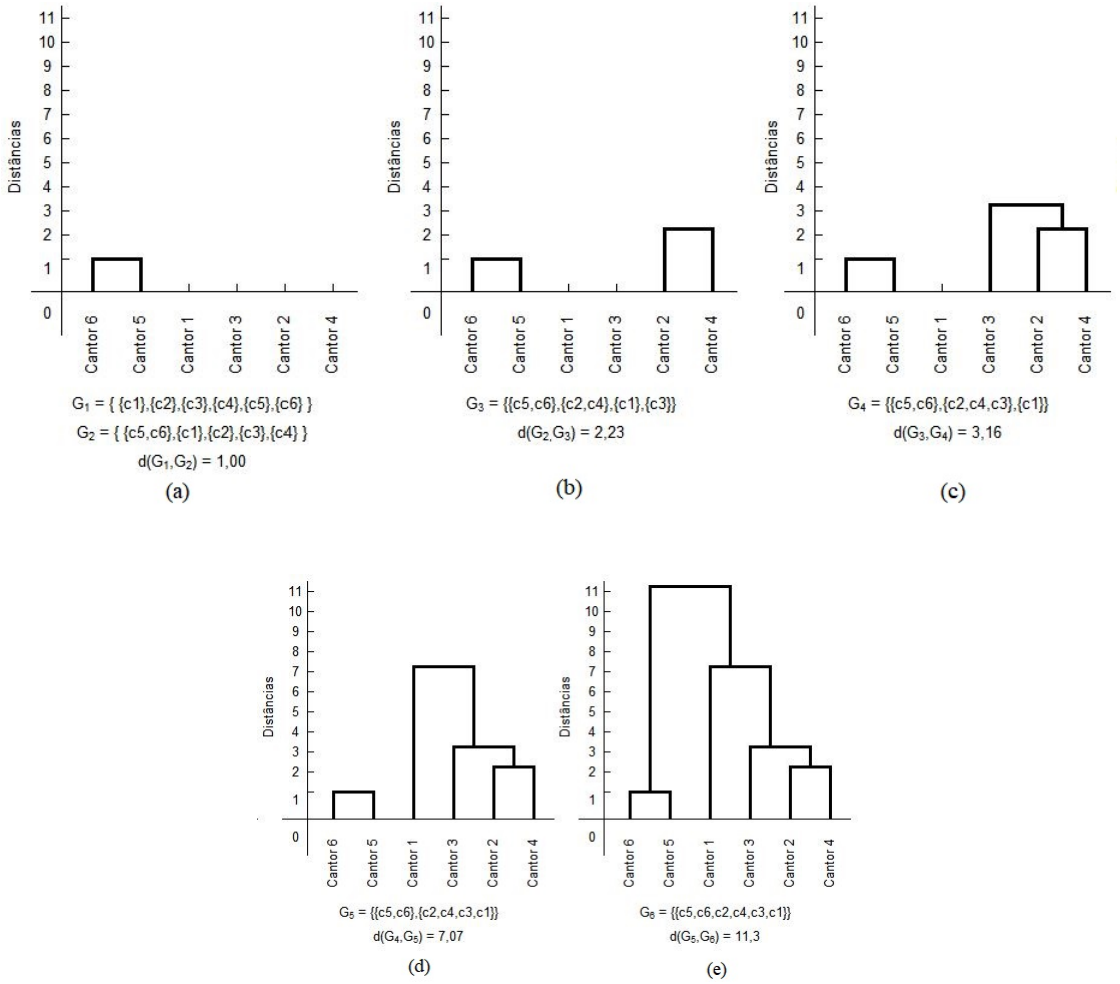
Outra maneira de representar graficamente a formação de agrupamentos é por meio de dendrograma. Hair *et al.* (2009) definem dendrograma como uma representação gráfica, em formato de árvore, que relaciona os objetos de estudo de maneira hierárquica.

Considerando os resultados obtidos na Tabela 4.4, cada objeto representa um Cantor da Tabela 4.1 (o objeto 1, representa o Cantor 1 (ou c1); o objeto 2, o Cantor 2 (ou c2); etc.). Sendo assim, as linhas da Tabela 4.4 também podem ser representadas como $G_1$ = {{c1},{c2}, {c3},{c4},{c5},{c6}}; $G_2$ = {{c5,c6},{c1},{c2},{c3},{c4}}; $G_3$ = {{c5,c6},{c2,c4},{c1}, {c3}}; $G_4$ ={{c5,c6}, {c2,c4,c3}, {c1}}; $G_5$ = {{c5,c6},{c2,c4,c3,c1}}; $G_6$ = {{c5,c6,c2,c4, c3,c1}} e as distâncias entre esses grupos correspondem a d($G_1$,$G_2$) = 1,00; d($G_2$,$G_3$) = 2,23; d($G_3$,$G_4$) = 3,16; d($G_4$,$G_5$) = 7,07; d($G_5$,$G_6$) = 11,3.

A construção do dendrograma começa a partir da identificação de quais objetos entre $G_1$ e $G_2$ se uniram para formar um grupo; essa união é representada na Figura 4.4(a) pela linha que une os Cantores 5 e 6. Essa linha crescerá verticalmente até a altura de 1,00 no eixo $y$ e representa a distância d($G_1$,$G_2$). A próxima representação será entre $G_2$ e $G_3$; seguindo os critérios descritos anteriormente, obtêm-se a representação na Figura 4.4(b); à medida que essa iteração vai acontecendo, o dendrograma vai representando, hierarquicamente,

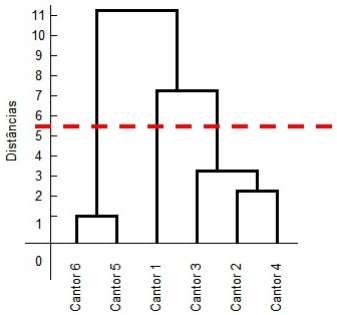
as relações entre os objetos.

Figura 4.4: Exemplo de construção do dendrograma.




Para identificar as formações de grupos possíveis, é necessário traçar uma linha horizontal sobre as hierarquias formadas no dendrograma. O número de grupos é determinado pelo número de interseções feitas pelas linhas horizontais e as linhas que determinam as hierarquias no dendrograma.

Figura 4.5: Identificação de grupos no dendrograma.



Na Figura 4.5, a linha tracejada em vermelho cruza três pontos diferentes na hierarquia do dendrograma identificando a seguinte formação de grupos $G_3$ = { {1},{2,3,4},{5,6} }. À medida que essas linhas horizontais se aproximam dos grupos unitários, a homogeneidade entre os objetos aumenta, ou seja, maior a probabilidade dos objetos do mesmo grupo compartilharem atributos semelhantes.

O termo *partição* na análise de agrupamento é empregado para formalizar um conjunto de grupos. Uma *partição* pode ser usada para identificar quais são as métricas de distância entre objetos e grupos que melhor representem os grupos de uma determinada tabela. Por exemplo, supondo que o objetivo da construção de um dendrograma seja agrupar dois tipos diferentes de cantores, se os cantores 5 e 6 são profissionais e os demais são amadores, uma *partição* que representa esse cenário é P = {{5,6},{1,2,3,4}}. As melhores escolhas dos métodos de cálculo de distância intragrupos ou intergrupos dependerá de qual delas vai gerar um conjunto G igual ou próximo de P. O quadro 4.2 exibe as métricas aplicadas e as formações de grupos sugeridas.

Quadro 4.2: Comparação entre métodos intragrupos e intergrupos.

| **Distâncias Intragrupo** | **Distâncias Intergrupo** | **Grupos** |
|---|---|---|
| Euclidiana | mínima | {1},{2,3,4,5,6} |
| Euclidiana | máxima | {1,2,3,4},{5,6} |
| Euclidiana | média | {1},{2,3,4,5,6} |
| Manhattan | mínima | {1},{2,3,4,5,6} |
| Manhattan | máxima | {1,2,3,4},{5,6} |
| Manhattan | média | {1},{2,3,4,5,6} |
| Canberra | mínima | {1},{2,3,4,5,6} |
| Canberra | máxima | {1},{2,3,4,5,6} |
| Canberra | média | {1},{2,3,4,5,6} |

Fonte: O autor.

As configurações que exibiram G igual a P foram as que usaram as *distâncias euclidianas ou Manhattan* (para calcular as distâncias entre os objetos) acompanhadas da *distância média* ( para calcular as distâncias entre grupos). Se forem respeitadas essas configurações e os atributos da Tabela 4.1, no próximo concurso de canto, o dendrograma irá sugerir dois grupos sem precisar recorrer a P. O dendrograma não vai identificar qual dos grupos contêm os melhores cantores, mas vai sugerir dois grupos e cada um deles reunirá os objetos que mais compartilham atributos.

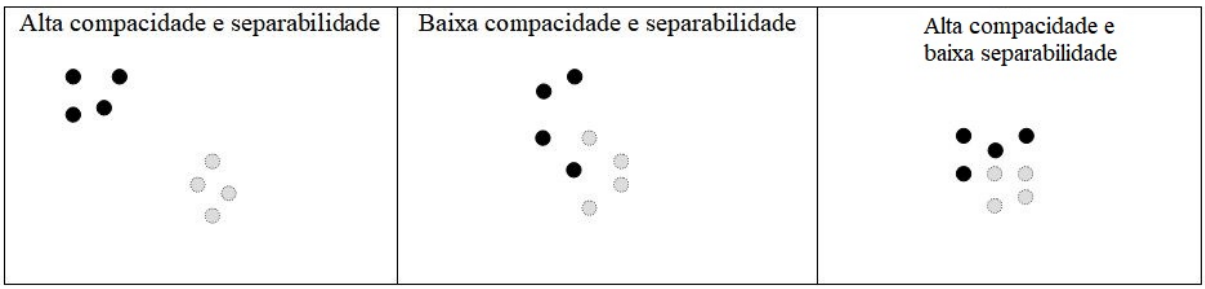
A análise de agrupamentos exige dos pesquisadores algumas precauções. Hair *et al.* (2009) sugerem que o processo de decisão em análises de agrupamentos deve passar por seis estágios: (i) definir os objetivos da análise de agrupamentos: o pesquisador deve estar atento aos objetivos de taxonomia dos grupos de objetos e lembrar que essa análise pode ser uma técnica exploratória e/ou confirmatória para identificar relações de similaridades

ou diferenças não detectadas; (ii) comparar o projeto de pesquisa aos resultados da análise de agrupamentos: o pesquisador deve avaliar se há observações atípicas, deve avaliar como medir as similaridades entre os objetos e se os dados precisam ser padronizados; (iii) avaliar as suposições na análise de agrupamentos: o pesquisador deve garantir que a amostra é representativa e que os resultados são generalizáveis para a população de interesse; (iv) determinar os agrupamentos e avaliar os ajustes gerais: o pesquisador escolhe as configurações do método de agrupamento, testa e reajusta os métodos; (v) interpretar os agrupamentos: a interpretação dos agrupamentos ajuda na confirmação das propostas e experiências; (vi) validar e definir os perfis de agrupamentos: nessa etapa o pesquisador deve validar ou refutar, a partir das amostras pesquisadas, o mesmo comportamento para uma população maior. Um dos focos da análise do agrupamento está em identificar as características dos objetos que contribuem para a formação dos grupos identificados.

## *4.2 Validação da análise de agrupamentos*

Validação é o processo de avaliação de resultados oriundos de análise de agrupamento. O objetivo é verificar se o resultado da análise de agrupamento pode ser identificado nos conjuntos de dados validados. Para validar, Halkidi, Batistakis & Vazirgiannis (2015) sugerem a avaliação/análise de dois itens: (i) compacidade: analisa a similaridades dos objetos dentro dos grupos (quanto menor as variâncias entre os objetos no grupo mais similares serão os objetos); (ii) separabilidade: analisa a dissimilaridade entre os grupos (quanto maior for a distância entre os grupos, mais diferentes eles serão). A Figura 4.6 exibe alguns exemplos de compacidade e separabilidade.

Figura 4.6: Comparabilidade e separabilidade.




Fonte: Adaptado de Silva, Peres & Boscarioli (2016)

As avaliações de compacidade e separabilidade podem ser feitas por meio de dois índices: (i) índices baseados em critérios externos; (ii) índices baseados em critérios internos. Para uma validação baseada em critérios externos, o pesquisador precisa conhecer a estrutura dos dados em relação à formação de grupos e em relação ao algoritmo (SILVA; PERES; BOSCARIOLI, 2016). Tendo em vista que a formação dos grupos já é conhecida, o objetivo de usar índices externos é identificar o quão eficiente é o algoritmo no reconhecimento dos grupos proposto pelo pesquisador. Essas avaliações podem ser calculadas por meio

de diferentes índices. Silva, Peres & Boscarioli (2016) sugerem a aplicação dos índices de *Rand, Jaccard ou Fowlkes & Mallows*. Esses índices são de fácil implantação, exigem menos processamentos que outros índices e analisam diferentes combinações entre o conjunto de grupos conhecido P e o conjunto gerado pela análise de agrupamento G.

Antes de calcular os índices de *Rand, Jaccard ou Fowlkes & Mallows*, é preciso analisar quais pares de objetos são encontrados nos conjuntos G e P e quais pares de objetos não são. Por exemplo, suponha que uma companhia aérea queira automatizar o processo de separação de malas fora do padrão e malas dentro de padrão; sendo {a,b,c,d} diferentes modelos de mala, a companhia sabe que a partição P = { {a,b,d}, {c} } representa um bom exemplo de organização, pois o primeiro grupo contêm as malas que estão no padrão e o segundo não, no entanto, o equipamento responsável por separar as malas gerou o seguinte conjunto G = { {a,d,c}, {b} }; o Quadro 4.3 exibe como são calculadas as comparações entre G e P.

Quadro 4.3: Combinação de pares de malas entre os conjuntos P e G.

| **Combinações entre G e P** | **ab** | **ac** | **ad** | **bc** | **bd** | **cd** | **Total** |
|---|---|---|---|---|---|---|---|
| (A) Pares que aparecem em G e P | | | x | | | | 1 |
| (B) Pares que aparecem em G e não em P | | x | | | | x | 2 |
| (C) Pares que aparecem em P e não em G | x | | | | x | | 2 |
| (D) Pares que não aparecem em P e em G | | | | x | | | 1 |

Fonte: adaptado de Silva, Peres & Boscarioli (2016).

Os totais de (A), (B), (C) e (D) são importantes para os cálculos dos índices de *Rand, Jaccard e Fowlkes & Mallows*. O índice de *Rand*, na Equação 4.8, diferentemente dos índices de *Jaccard ou Fowlkes & Mallows*, dá a mesma importância para as combinações (A) e (D); é como se os pares de objetos que não aparecem entre G e P tivesse a mesma importância que os pares de objetos que aparecem entre G e P.

$$I_{Rand} = \frac{(A + D)}{(A + B + C + D)} \tag{4.8}$$

O índice de *Rand* aplicado aos conjuntos G e P seria $I_{Rand} = (1+1)/(1+2+2+1) = 0{,}333$. Quanto mais próximo o índice estiver do valor 1, mais semelhantes serão G e P, e quanto mais próximo de 0, menos semelhantes G e P serão.

O índice de *Jaccard*, na Equação 4.9, considera somente os pares de objetos que aparecem nos conjuntos G e P. Os valores gerados na linha (D) (i.e., pares de objetos não encontrados nos conjuntos G e P) não influenciam o índice.

$$I_{Jaccard} = \frac{A}{(A+B+C)} \tag{4.9}$$

O índice de *Jaccard* aplicado em G e P gera $I_{Jaccard} = (1/(1+2+2)) = 0{,}2$. No índice de *Jaccard*, os valores próximos de 0 sugerem pouca semelhança entre G e P e os valores próximos de 1 sugerem muita semelhança.

A escala de valores do índice de *Fowlkes & Mallows* na Equação 4.10 também varia de 0 a 1 (0 para sugerir pouca semelhança entre os conjuntos G e P e 1 para sugerir muita semelhança). A diferença entre a escala de resultados do índice de *Fowlkes & Mallows* e as de *Rand* e de *Jaccard* é que, quando os conjuntos G e P são realmente diferentes, o índice de *Fowlkes & Mallows* gera valores mais próximos de 0 se comparado aos outros dois.

$$I_{FM} = \sqrt{\frac{A}{(A+B)} * \frac{A}{(A+C)}} \tag{4.10}$$

Aplicando o índice de *Fowlkes & Mallows*, na Equação 4.10, aos conjuntos G e P, obtém-se $I_{FM} = \sqrt{(1/(1+2)) * (1/(1+2))} = 0,333$. Os índices de *Rand, Jaccard ou Fowlkes & Mallows* consideram diferentes cenários permitindo ao pesquisador optar por aquele que atenda às características de sua pesquisa ou optar por mais de um, caso queira avaliar as formações de grupos por diferentes perspectivas.

O Algoritmo 4.11 calcula os índices de *Rand, Jaccard* e *Fowlkes & Mallows*. Ele recebe como parâmetro de entrada um vetor com os grupos a serem avaliados (i.e., o conjunto G); recebe um vetor com grupos conhecidos (i.e., o conjunto P); e um valor entre 1 e 3 que representa, respectivamente, os índices de *Rand, Jaccard ou Fowlkes & Mallows*.

**Algoritmo 4.11** Calcular os índices baseados em critérios externos *Rand, Jaccard ou Fowlkes & Mallows*

```
1:  Método IndicesExternos(G,P,indice)
2:      v1 ← Vetor[Tamanho(G)]
3:      v1 ← GerarSequenciaDeNumeros(De : 1Ate : Tamanho(G))
4:      m1 ← ListarParesDeGrupos(v1)
5:      m2 ← Matriz[Tamanho(G)][6]
6:      Para i ← 1 até TotalDeLinhas(m1) Faça
7:          m2[i][1] < −m1[i][1]
8:          m2[i][2] < −m1[i][2]
9:          m2[i][3] < −m2[i][4] < −m2[i][5] < −m2[i][6] < −0
10:     FimPara
```

```
11:     Para i ← 1 até TotalDeLinhas(m2) Faça
12:         Se (P[m2[i][1]]=P[m2[i][2]]) E (G[m2[i][1]]=G[m2[i][2]]) E (G[m2[i][1]]=P[m2[i][1]]) E
    (G[m2[i][2]]=P[m2[i][2]]) Então
13:             m2[i][3] ← 1
14:         FimSe
15:         Se (G[m2[i][1]] = G[m2[i][2]]) E (P[m2[i][1]] != P[m2[i][2]]) Então
16:             m2[i][4] ← 1
17:         FimSe
18:         Se (G[m2[i][1]] != G[m2[i][2]]) E (P[m2[i][1]] = P[m2[i][2]]) Então
19:             m2[i][5] ← 1
20:         FimSe
21:         Se (G[m2[i][1]] != G[m2[i][2]]) E (P[m2[i][1]] != P[m2[i][2]]) Então
22:             m2[i][6] ← 1
23:         FimSe
24:     FimPara
25:     a ← b ← c ← d ← 0
26:     Para i ← 1 até TotalDeLinhas(m2) Faça
27:         a < −a + m2[i][3]
28:         b < −b + m2[i][4]
29:         c < −c + m2[i][5]
30:         d < −d + m2[i][6]
31:     FimPara
32:     Se (indice = 1) Então
33:         indice ← (a + d)/(a + b + c + d)
34:     FimSe
35:     Se (indice = 2) Então
36:         indice ← a/(a + b + c)
37:     FimSe
38:     Se (indice = 3) Então
39:         indice ← RaizQuadrada((a/(a + b)) ∗ (a/(a + c)))
40:     FimSe
41:     Retorna indice
42: Fim Método
```

O método *IndicesExternos()*, no Algoritmo 4.11 linha 1, cria uma tabela similar ao Quadro 4.3. Esse métodos calcula, nas linhas 32, 35 e 38, respetivamente, os índices baseados em critérios externos de *Rand, Jaccard ou Fowlkes & Mallows* e retorna, na linha 41, o valor do índice.

As avaliações baseadas em critérios internos se diferenciam das avaliações baseadas em critérios externos por não dependerem de informações extras sobre outros conjuntos de

grupos. Para a avaliação baseada em critérios internos, Silva, Peres & Boscarioli (2016) sugerem a aplicação dos índices de *Dunn, Davis-Bouldin e/ou Silhouette.*

O índice de *Dunn* considera em seus cálculos as distâncias entre diferentes grupos e o tamanho de cada grupo. Partindo de zero, quanto maior for o resultado do índice de *Dunn*, melhor a separabilidade e compacidade de grupos (e.g., Figura 4.6).

$$I_{Dunn} = min_{1 \leqslant i \leqslant k} \left\{ min_{1 \leqslant j \leqslant k, i \neq j} \left\{ \frac{dist(g_i, g_j)}{max(disp(g_k))} \right\} \right\} \tag{4.11}$$

Na Equação 4.11, *dist($g_i$,$g_j$)* representa a distância entre dois grupos (que corresponde à menor distância euclidiana existente entre os objetos do primeiro grupo e os objetos do segundo); *disp($g_k$)* representa a medida da dispersão (i.e., a maior distância entre objetos no mesmo grupo); *i,j* são pares no conjunto de grupos de {1,2,..,$k$}, sendo $i \neq j$; e $k$ é o total de grupos no conjunto pesquisado. Dentro da função *dist($g_i$,$g_j$)* é possível optar pelas estratégias de *menor distância*, *maior distância* ou *distância média* (e.g., Quadro 4.1). Dentro da função *disp($g_k$)* é possível optar pelas estratégias *maior distância* ou *distância média* entre os objetos.

Aplicando o índice de *Dunn* ao conjunto P = {1,1,1,1,2,2}, que separa os Cantores da Tabela 4.1 divididos em dois grupos, primeiro obtém-se a menor distância entre todos os pares de grupos possíveis, no caso, *dist($g_1$,$g_2$)*=11,3137 (esse valor está na Tabela 4.2 entre os objetos 1 e 5 e equivale a estratégia de *maior distância* entre grupos) e depois a maior dispersão entre todos os grupos, nesse caso, *disp($g_1$)*=7,0710 (na Tabela 4.2 o valor aparece entre os objetos 1 e 2, a maior distância entre objetos do mesmo grupo e equivale à estratégia de *maior distância* entre objetos). Com essas informações, o índice de *Dunn* é $I_{Dunn}$ = (11,3137/7,0710) = 1,6. Como o conjunto P contém uma formação de grupo conhecida e válida, esse resultado de 1,6 servirá de parâmetro para os próximos concursos de canto; se o próximo conjunto a validar obtiver um índice de *Dunn* inferior a 1,6, menos precisa será a formação desses grupos; e se o índice for maior que 1,6, mais precisa foi a formação dos grupos.

O primeiro algoritmo a ser considerado para calcular a distância de *Dunn* é o Algoritmo 4.12, que calcula a *menor distância, maior distância* ou a *distância média* entre grupos.

**Algoritmo 4.12** Calcular as distâncias entre grupos considerando a *menor distância, maior distânca* ou a *distância média*

```
1: Método DistanciaEntreGrupos(tabela,G,met_intergrupos)
2:     Método GravarDist(matriz,g1,g2, valor)
3:         Para i ← 1 até TotalDeLinhas(matriz) Faça
4:             Se (matriz[i][1]=g1 E matriz[i][2]=g2) Ou (matriz[i][1]=g2 E matriz[i][2]=g1) Então
5:                 matriz[i][3] ← valor
6:                 matriz[i][4] ← matriz[i][4] + 1
7:                 break
8:             FimSe
9:         FimPara
10:        Retorna matriz
11:    Fim Método
12:    distEucl ← Calculo_De_Distancias(tabela, 1)
13:    vg ← ListarGrupos(G)
14:    pg ← ListarParesDeGrupos(vg)
15:    m ← Matriz[TotalDeLinhas(pg)][4]
16:    Para i ← 1 até TotalDeLinhas(pg) Faça
17:        m[i][1] ← pg[i, 1]
18:        m[i][2] ← pg[i, 2]
19:        m[i][3] ← 0
20:        m[i][4] ← 0
21:    FimPara
22:    Para i ← 1 até TotalDeLinhas(distEucl) Faça
23:        obj01 ← distEucl[i][1]
24:        obj02 ← distEucl[i][2]
25:        Se (G[obj01] != G[obj02]) Então
26:            distancia ← DistanciaEntre_Dois_Objetos(G[obj01], G[obj02], m)
27:            Se (met_intergrupos=1) Então
28:                Se distancia=0 Ou distEucl[i][3] < distancia Então
29:                    m ← GravarDist(m, G[obj01], G[obj02], distEucl[i][3])
30:                FimSe
31:            FimSe
32:            Se met_intergrupos=2 Então
33:                Se distEucl[i][3] > distancia Então
34:                    m ← GravarDist(m, G[obj01], G[obj02], distEucl[i][3])
35:                FimSe
36:            FimSe
37:            Se met_intergrupos=3 Então
38:                distancia ← distancia + distEucl[i][3]
39:                m ← GravarDist(m, G[obj01], G[obj02], distancia)
40:            FimSe
```

```
41:         FimSe
42:     FimPara
43:     Se met_intergrupos=3 Então
44:         Para i ← 1 até TotalDeLinhas(m) Faça
45:             m[i][3] ← (m[i][3]/m[i][4])
46:         FimPara
47:     FimSe
48:     Retorna indice
49: Fim Método
```

O Método *DistanciaEntreGrupos(tabela,G,met_intragrupo)*, no Algoritmo 4.12 linha 1, recebe uma tabela (e.g., Tabela 4.1); um vetor identificando grupos e objeto (e.g. G={1,1,1, 1,2,2}); e um valor de 1 a 3 (1 - Representa o cálculo de *menor distância*; 2 - *Maior Distância; 3 - Distâncias Médias*; conforme ilustrações no Quadro 4.1). Como parâmetro de saída, o método *DistanciaEntreGrupos* retorna uma tabela com as combinações de grupos possíveis, as distâncias entre esses pares de grupos e o número de distâncias identificadas pelos cálculos de *médias, maiores ou menores distâncias*. Por exemplo, se G = {1,1,2,2,3,3}, *met_intergrupos*=3 e tabela = Tabela 4.1, o método *DistanciaEntreGrupos(tabela,G,met_intergrupos)* retornaria os valores conforme a Tabela 4.5.

Tabela 4.5: Distâncias entre Grupos

| Grupo01 | Grupo02 | Distância | Quantidade |
|---|---|---|---|
| 1 | 2 | 4,144671 | 4 |
| 1 | 3 | 7,448012 | 4 |
| 2 | 3 | 5,867882 | 4 |

Fonte: O autor.

O próximo algoritmo que apoiará o cálculo do índice de *Dunn* é o Algoritimo 4.13, que recebe como parâmetro de entrada uma tabela com os objetos e seus atributos (e.g., Tabela 4.1); um vetor representando grupos e objetos (e.g., G = {1,1,2,2,3,3}); e um valor numérico com valor igual a 1 ou 2 (1 - representa a maior distância entre dois objetos no mesmo grupo; 2 - representa a distância média de todos os objetos no grupo). O método *DispersaoIntraGrupos()*, no Algoritmo linha1, 4.13 retorna uma tabela com as dispersões de cada grupo e a quantidade de distâncias consideradas.

**Algoritmo 4.13** Calcular as dispersões nos grupos considerando a *maior distânca* ou a *distância média*

```
1: Método DispersaoIntraGrupos(tabela,G,met_intragrupo)
2:     distEucl ← Calculo_De_Distancias(tabela, 1)
3:     vg ← ListarGrupos(G)
4:     m ← Matriz[Tamanho(vg)][2]
5:     m[][] ← 0
6:     Para i ← 1 até TotalDeLinhas(distEucl) Faça
7:         obj01 ← distEucl[i][1]
8:         obj02 ← distEucl[i][2]
9:         Se G[obj01] = G[obj02] Então
10:            m[G[obj01]][2] ← m[G[obj01]][2] + 1
11:            Se (met_intragrupo=1) E (m[G[obj01]][1]<distEucl[i][3]) Então
12:                m[G[obj01]][1] ← distEucl[i][3]
13:            FimSe
14:            Se met_intragrupo=2 Então
15:                m[G[obj01]][1] ← m[G[obj01]][1] + distEucl[i][3]
16:            FimSe
17:        FimSe
18:    FimPara
19:    Se met_intragrupo=2 Então
20:        Para i ← 1 até TotalDeLinhas(m) Faça
21:            Se m[i][1]!=0 Então
22:                m[i][1] ← (m[i][1]/m[i][2])
23:            FimSe
24:        FimPara
25:    FimSe
26:    Retorna m
27: Fim Método
```

Se o método *DispersaoIntraGrupos(tabela,G,met_intragrupo)* receber a *tabela*=Tabela 4.1, $G$ = {1,1,2,3,3,3} e *met_intragrupo*=2, o método irá retornar os valores conforme a Tabela 4.6.

Tabela 4.6: Dispersão dos Grupos

| | Dispersão | Quantidade |
|---|---|---|
| 1. | 7,071068 | 1 |
| 2. | 0,000000 | 0 |
| 3. | 3,619100 | 3 |

Fonte: O autor.

A linha 1 da Tabela 4.6 representa o grupo 1 do conjunto G, a distância média entre todos os objetos do grupo 1 é 7,071 e existe um par de combinações entre os objetos desse grupo. A linha 2 representa o grupo 2; a distância média entre os objetos do grupo 2 é 0 porque existe somente um objeto nesse grupo. A linha três representa o grupo 3, a distância média entre todos os objetos é 3,6191 e há três combinações possíveis de pares de objetos nesse grupo.

Os Algoritmos 4.12 e 4.13 serão úteis para calcular as funções *dist($g_i$,$g_j$)* e *disp($g_k$)* da Equação 4.11. O Algoritmo 4.14 é responsável pelo cálculo do índice de *Dunn*.

**Algoritmo 4.14** Calcular o índice de *Dunn*

```
1:  Método IndiceDeDunn(tabela,G,met_intragrupo,met_intergrupos)
2:      distGG ← dispGk ← indice ← 0
3:      vg ← ListarGrupos(G)
4:      pg ← ListarParesDeGrupos(vg)
5:      m1 ← Matriz[TotalDeLinhas(pg)][4]
6:      m2 ← Matriz[Tamanho(vg)][2]
7:      m1 ← DistanciaEntreGrupos(tabela, G, met_intergrupos)
8:      m2 ← DispersaoIntraGrupos(tabela, G, met_intragrupo)
9:      Para i ← 1 até TotalDeLinhas(m1) Faça
10:         Se m1[i][3] > 0 Então
11:             Se (distGG=0) Ou (m1[i][3]<distGG) Então
12:                 distGG ← m1[i][3]
13:             FimSe
14:         FimSe
15:     FimPara
16:     Para i ← 1 até TotalDeLinhas(m2) Faça
17:         Se m2[i][1]>0 Então
18:             Se (dispGk=0) Ou (m2[i][1]>dispGk) Então
19:                 dispGk ← m2[i, 1]
20:             FimSe
21:         FimSe
22:     FimPara
23:     Retorna (distGG/dispGk)
24: Fim Método
```

Se o método *IndiceDeDunn(tabela,G,met_intragrupo,met_intergrupos)* receber $G = \{1, 1, 1, 1, 2, 2\}$, *tabela* = Tabela 4.1, *met_intragrupo* = 1 (i.e., *maior distância* entre objetos), *met_intergrupos* = 2 (i.e., *maior distância* entre grupos.), o resultado será 1,6.

O índice de *Davies-Bouldin*, na Equação 4.12, usa a dispersão de cada grupo e as distâncias entre as combinações de grupos para inferir uma semelhança entre eles. Enquanto o índice

de *Dunn* olha, em termos gerais, para um valor mínimo na relação distância entre grupos e dispersão máxima nos grupos, o índice de *Davies-Bouldin* olha a média nas similaridades entre a relação da dispersão dos grupos e a distância entre eles, dessa forma, o índice favorece a validação de grupos com objetos muito similares entre si.

$$I_{DB} = \frac{1}{k}\sum_{i=1}^{k} \max_{i \neq j} \left\{ \frac{disp(g_i) + disp(g_j)}{dist(g_i, g_j)} \right\} \tag{4.12}$$

Quanto mais próximo de zero for o resultado do índice de *Davies-Bouldin*, mais bem formados serão os grupos. O termo $k$ na Equação 4.12 representa o total de grupos a serem analisados; $dist(g_i,g_j)$ representa a distância entre dois grupos distintos (internamente, nessa função, usam-se as estratégias de *menor distância*; *maior distância* ou *distância média* (e.g., Quadro 4.1)) e $disp(g_i)$ e $disp(g_j)$ representam as dispersões nos grupos $g_i$ e $g_j$ (internamente, na função *disp()*, usam-se as estratégias de *maior distância* ou *distância média* entre objetos). Por exemplo, um conjunto de grupos válidos para Tabela 4.1 é G = {1,1,1,1,2,2}, calculando as distâncias euclidianas entre os objetos, a distância entre $g_1$ e $g_2$ $dist(g_1,g_2)$ seria 11,313, a dispersão de $g_1$ e $g_2$ seria $disp(g_1)$=7,071 e $disp(g_2)$=1,0, e o índice de *Davies-Bouldin* ficaria $I_{DB} = (1/2) * ((7,017 + 1)/(11,313) + (7,017 + 1)/(11,313)) = 0,713$. A função *max*, na Equação 4.12, retorna a maior relação $((disp(g_i)+disp(g_j))/\text{dist}(g_{i,j})$ de que cada grupo $g$ participa, ou seja, o valor máximo dessa relação em $g_1$ é (7,017+1)/(11,313) e em $g_2$ é (7,017+1)/(11,313), por isso, na solução do $I_{DB}$, o cálculo (7,017+1)/(11,313) aparace duas vezes. O Algoritmo 4.15 é usado para calcular o índice de *Davies-Bouldin*.

**Algoritmo 4.15** Calcular o índice de *Davies-Bouldin*

```
1:  Método IndiceDeDaviesBouldin(tabela,G,met_intragrupo,met_intergrupos)
2:      distGG ← dispGk ← indice ← 0
3:      vg ← ListarGrupos(G)
4:      pg ← ListarParesDeGrupos(vg)
5:      m1 ← Matriz[TotalDeLinhas(pg)][4]
6:      m2 ← Matriz[Tamanho(vg)][2]
7:      k ← Tamanho(vg)
8:      m1 ← DistanciaEntreGrupos(tabela, G, met_intergrupos)
9:      m2 ← DispersaoIntraGrupos(tabela, G, met_intragrupo)
10:     m3 ← Matriz[TotalDeLinhas(pg)][6]
11:     m3[][] ← 0
```

```
12:     Para i ← 1 até TotalDeLinhas(m1) Faça
13:         m3[i][1] ← m1[i][1]
14:         m3[i][2] ← m1[i][2]
15:         m3[i][3] ← m2[(m1[i][1])][1]
16:         m3[i][4] ← m2[(m1[i][2])][1]
17:         m3[i][5] ← m1[i][3]
18:         Se (m3[i][3]+m3[i][4]=0) Ou (m3[i][5]=0) Então
19:             m3[i][6] ← 0
20:         Senão
21:             m3[i][6] ← ((m3[i][3] + m3[i][4])/m3[i][5])
22:         FimSe
23:     FimPara
24:     RelGG ← valor < −0
25:     Para i ← 1 até k Faça
26:         valor ← 0
27:         Para j ← 1 até TotalDeLinhas(m3) Faça
28:             Se m3[j][1]= i Ou m3[j][2]=i Então
29:                 Se m3[j][6] > valor Então
30:                     valor ← m3[j, 6]
31:                 FimSe
32:             FimSe
33:         FimPara
34:         RelGG ← RelGG + valor
35:     FimPara
36:     Retorna ((1/k) ∗ RelGG)
37: Fim Método
```

O método *IndiceDeDaviesBouldin(tabela,G,met_ intragrupo,met_ intergrupos)* recebe os mesmos parâmetros de entrada vistos no método *IndiceDeDunn(tabela, G, met_ intragrupo, met_ intergrupos))* e retorna o valor do índice de *Davies-Bouldin*. Entre as linhas 12 e 23, no Algoritmo 4.15, é construída uma tabela de apoio; supondo o conjunto G={1,1,1,3,2,2}, essa tabela se configuraria conforme a Tabela 4.7.

Tabela 4.7: Tabela de apoio do índice de Davies-Bouldin

| $g_1$ | $g_2$ | disp($g_1$) | disp($g_2$) | dist($g_1$,$g_2$) | $\frac{disp(g_1)+disp(g_2)}{dist(g_1,g_2)}$ |
|---|---|---|---|---|---|
| 1 | 3 | 7,071 | 0 | 6,708 | 1,0540 |
| 1 | 2 | 7,071 | 1 | 11,313 | 0,7133 |
| 3 | 2 | 0,000 | 1 | 5,385 | 0,1856 |

Fonte: O autor.

As duas primeiras colunas da Tabela 4.7 representam a combinação de pares de grupos possíveis; *disp(g₁)* e *disp(g₂)* são, respectivamente, as dispersões dos grupos $g_1$ e $g_2$; dist($g_1$, $g_2$) representa a distância entre $g_1$ e $g_2$; e a equação $(disp(g_1) + disp(g_2)/dist(g_1, g_2))$ representa a semelhança entre os pares de grupos $g_1$ e $g_2$. O laço de repetição que começa na linha 25 do Algoritmo 4.15 busca e armazena, na variável *RelGG*, os maiores resultados em que participam $g_1$, $g_2$ e $g_3$. O índice de *Davies-Bouldin* com G={1,1,1,3,2,2} seria $I_{DB} = (1/3) * (1,0540 + 0,7133 + 1,0540) = 0,9405$. Quanto menor o índice de *Davies-Bouldin*, melhor é a formação dos grupos. O índice de *Davies-Bouldin* para G={1,1,1,1,2,2} foi 0,713 e para G={1,1,1,3,2,2} foi 0,940; ao comparar esses resultados a melhor formação de grupos é G={1,1,1,1,2,2}.

O índice de *Silhouette* foi idealizado, inicialmente, como uma forma de visualizar o quanto cada objeto é parecido com os outros objetos do seu grupo. Esse índice gera valores que vão de -1 a +1. Quanto mais próximo o índice for de +1, melhor é a formação dos grupos; quanto mais próximo de -1, pior é a formação dos grupos.

$$I_{Sil} = \frac{1}{k} \sum_{1}^{k} \left\{ \frac{b(i) - a(i)}{max\{a(i), b(i)\}} \right\} \tag{4.13}$$

Na Equação 4.13, $k$ representa o total de objetos pesquisados; *a(i)* representa a distância média de um objeto $i$ em relação aos demais objetos do grupo a que $i$ pertence; *b(i)* representa a distância média entre o objeto $i$ e todos os objetos dos grupos mais próximos ao grupo de $i$. O Algoritmo 4.16 calcula o índice de Silhouette.

**Algoritmo 4.16** Calcular o índice de *Silhouette*

```
1:  Método IndiceDeSilhouette(tabela, G)
2:      n ← TotalDeLinhas(tabela)
3:      vg ← ListarGrupos(G)
4:      m1 ← Matriz[Tamanho(vg)][4]
5:      m1[][] ← 0
6:      m2 ← Matriz[n][9]
7:      m2[][] ← 0
8:      grupoprox ← menordist ← paresEmG ← tmp ← 0
9:      Para i ← i até n Faça
10:         menordist ← paresEmG ← grupoprox ← 0
11:         m1[][] ← 0
```

```
12:     Para j ← 1 até Tamanho(G) Faça
13:         Se G[i]!=G[j] Então
14:             V1 ← Vetor(tabela[i][2 : TotalDeColunas(tabela)])
15:             V2 ← Vetor(tabela[j][2 : TotalDeColunas(tabela)])
16:             tmp ← Eq_Euclidiana(V1, V2)
17:             m1[G[j]][1] ← m1[G[j]][1] + tmp
18:             m1[G[j]][2] ← m1[G[j]][2] + 1
19:             Se m1[G[j]][1]!=0 E m1[G[j]][2]!=0 Então
20:                 m1[G[j]][3] ← m1[G[j]][1]/m1[G[j]][2]
21:             FimSe
22:         FimSe
23:         m1[G[j]][4] ← m1[G[j]][4] + 1
24:     FimPara
25:     Para j ← 1 até TotalDeLinhas(m1) Faça
26:         Se m1[j][3] != 0 Então
27:             Se (menordist=0) Ou (m1[j][3]<menordist) Então
28:                 menordist < −m1[j][3]
29:                 grupoprox < −j
30:             FimSe
31:         FimSe
32:     FimPara
33:     m2[i][1] < −grupoprox
34:     m2[i][2] < −menordist
35:     m2[i][8] < −m1[G[i]][4]
36:     m2[i][9] < −G[i]
37: FimPara
38: Para i ← 1 até n Faça
39:     tmp ← 0
40:     Para j ← 1 até n Faça
41:         Se (m2[i][1]=G[j]) E (i != j) Então
42:             tmp ← tmp + 1
43:         FimSe
44:         Se (G[j]=G[i]) E (i != j) Então
45:             m2[i][3] ← m2[i][3] + 1
46:         FimSe
47:     FimPara
48:     m2[i][4] ← tmp
49: FimPara
```

50: $a \leftarrow b \leftarrow tot_obj \leftarrow 0$
51: **Para** i $\leftarrow$ 1 **até** Tamanho(G) **Faça**
52: $a \leftarrow b \leftarrow 0$
53: **Para** j $\leftarrow$ 1 **até** Tamanho(G) **Faça**
54: **Se** i!=j **Então**
55: $V1 \leftarrow Vetor(tabela[i][2 : TotalDeColunas(tabela)])$
56: $V2 \leftarrow Vetor(tabela[j][2 : TotalDeColunas(tabela)])$
57: **Se** G[i]=G[j] **Então**
58: $a \leftarrow a + Eq_Euclidiana(V1, V2)$
59: **FimSe**
60: **Se** G[i]!=G[j] E G[j]=m2[i][1] **Então**
61: $b \leftarrow b + Eq_Euclidiana(V1, V2)$
62: **FimSe**
63: **FimSe**
64: **FimPara**
65: **Se** (m2[i][3] != 0) E (a!=0) **Então**
66: $m2[i][5] \leftarrow a/m2[i][3]$
67: **FimSe**
68: **Se** (m2[i][4] != 0) E (b!=0) **Então**
69: $m2[i][6] \leftarrow b/m2[i][4]$
70: **FimSe**
71: **Se** m2[i][5]>m2[i][6] **Então**
72: $m2[i][7] \leftarrow (m2[i][6] - m2[i][5])/m2[i][5]$
73: **Senão**
74: $m2[i][7] \leftarrow (m2[i][6] - m2[i][5])/m2[i][6]$
75: **FimSe**
76: **Se** m2[i][8]=1 **Então**
77: $m2[i][7] \leftarrow 0$
78: **FimSe**
79: **FimPara**
80: $Isil \leftarrow 0$
81: **Para** i $\leftarrow$ 1 **até** TotalDeLinhas(m2) **Faça**
82: $Isil < -Isil + m2[i, 7]$
83: **FimPara**
84: $(Isil/n)$
85: **Fim Método**

Em termos gerais, o índice de *Silhouette* calcula $s(i) = (b(i) - a(i))/max(a(i), b(i))$ para cada objeto $i$ da tabela. O resultado do índice é a soma de $s(i)$ dividida pelo total de objetos. Quando um objeto $i$ é o único elemento de um grupo, o $s(i)$ recebe zero. Rousseeuw (1987) argumenta que, em grupo com um objeto, fica impreciso definir o valor

de *a(i)* e que atribuir zero ao $s(i)$ parece ser a forma mais neutra de trabalhar com esses casos.

O método *indiceDeSilhouette(tabela,G)* recebe como parâmetro de entrada uma tabela (e.g., Tabela 4.1) e um vetor de grupos e objetos (e.g., G={1,1,1,1,2,2}). Para exemplificar o cálculo do índice de *Silhouette*, considere G={1,2,2,2,3,3} e a *tabela* = Tabela 4.1. O laço de repetição, que começa na linha 9 do Algoritmo 4.16, criará uma matriz similar à Tabela 4.8.

Tabela 4.8: Tabela de apoio do índice de Silhouette

| i | Grupo Vizinho | Pares no Grupo | Pares entre os Grupos | a(i) | b(i) | s(i) | Objetos no Grupo | Grupo |
|---|---|---|---|---|---|---|---|---|
| 1. | 2 | 0 | 3 | 0,000 | 6,083 | 0,000 | 1 | 1 |
| 2. | 3 | 2 | 2 | 2,699 | 3,924 | 0,312 | 3 | 2 |
| 3. | 1 | 2 | 1 | 2,699 | 4,472 | 0,396 | 3 | 2 |
| 4. | 3 | 2 | 2 | 2,236 | 4,928 | 0,546 | 3 | 2 |
| 5. | 2 | 1 | 3 | 1,000 | 5,612 | 0,821 | 2 | 3 |
| 6. | 2 | 1 | 3 | 1,000 | 4,826 | 0,792 | 2 | 3 |

Fonte: O autor.

Na Tabela 4.8, a coluna $i$ representa cada objeto; a coluna *Grupo Vizinho* representa o grupo mais próximo de $i$, que é identificado calculando a menor média das distância entre $i$ e os objetos dos outros grupos; a coluna *Pares no Grupo* é o total de combinações de $i$ com os outros objetos do grupo de $i$; a coluna *Pares entre os Grupos* é o total de combinações de $i$ com os objetos do *Grupo Vizinho*; as colunas *a(i)*, *b(i)* e *s(i)* representam, respectivamente, a distância média de $i$ e seus colegas de grupo, a distância média de $i$ e os colegas do grupo mais próximo e o índice de *Silhouette* para o objeto $i$; a coluna *Objetos no Grupo* representa quantos objetos existem no grupo de que $i$ participa (inclusive $i$); e a coluna *Grupo* é a identificação do grupo a que $i$ pertence.

O objeto 1, na Tabela 4.8, tem o valor de *a(i)* igual a zero porque, por definição, quando um grupo tem só um objeto, o valor de *a(i)* e *s(i)* ficam igual a zero. As distâncias euclidianas do objeto 1 para todos os objetos do grupo 2 (*Grupo Vizinho*) são: 7,0710 (do objeto 1 para o 2); 4,472 (do objeto 1 para o 3); e 6,708 (do objeto 1 para o 4) (as distâncias entre os objetos estão na Tabela 4.2). O resultado de *b(1)* é a soma das distâncias entre o objeto 1 e todos os objetos do grupo mais próximo dele dividido pelo valor de *Pares entre os Grupos* (i.e., 3); *b(1)*= (7,0710 + 4,472 + 6,708)/ 3 = 6,082. Para o objeto 2 da Tabela 4.8, *a(2)=(2,236+3,162)/2=2,699*, *b(2)=(4,242+3,605)/2=3,924*, *s(2)=(3,924-2,699)/3,924=0,312*. Depois de calcular *s(i)* de todos os objetos, o índice de

*Silhoutte* é a somatória de *s(i)* dividido pelo total de objetos; $I_{Sil} = 2,869/6 = 0,4782$.

Rousseeuw (1987) argumenta que o índice *Silhouette* é mais útil quando as distâncias entre os objetos seguem uma escala proporcional, como as aferidas pelas distâncias euclidianas. Esse índice obtém melhores resultados quando os objetos com atributos semelhantes formam grupos com formato esférico, compacto e bem separados de outros grupos.

## *4.3 Escalonamento multidimensional*

O escalonamento multidimensional é uma técnica usada para projetar em um espaço multidimensional os objetos de estudo. Também conhecido como mapeamento perceptual, ele constrói um mapa cujos extremos representam características dos objetos e sobre esse mapa são projetados os objetos de acordo com suas características. Por exemplo, em um gráfico de coordenadas $xy$, o eixo $y$ representada uma escala de sabores do mais doce até o mais azedo e o eixo $x$ representada uma escala de valores do mais caro até o mais barato; se o pesquisador estiver interessado em saber se o sabor tem influência sobre o preço dos alimentos, ele pode projetar, nesse gráfico, os alimentos pesquisados e analisar os resultados.

Nesse tipo de representação, quanto mais próximos estiverem os objetos, mais semelhantes entre si eles serão. Segundo Hair *et al.* (2009), o escalonamento multidimensional é uma técnica exploratória usada para obtenção de avaliações comparativas entre objetos de forma a identificar comportamentos não observados em outras análises. Para os autores, é importante que o pesquisador: (i) saiba escolher os objetos de estudo; (ii) saiba que esse método é usado para analisar os dados por similaridade ou preferências do respondente; (iii) opte por analisar os objetos individualmente ou em grupo. Usar esse método para outras finalidades pode gerar erros de interpretação.

No Quadro 4.4, segue um exemplo de aplicação do escalonamento multidimensional em R.

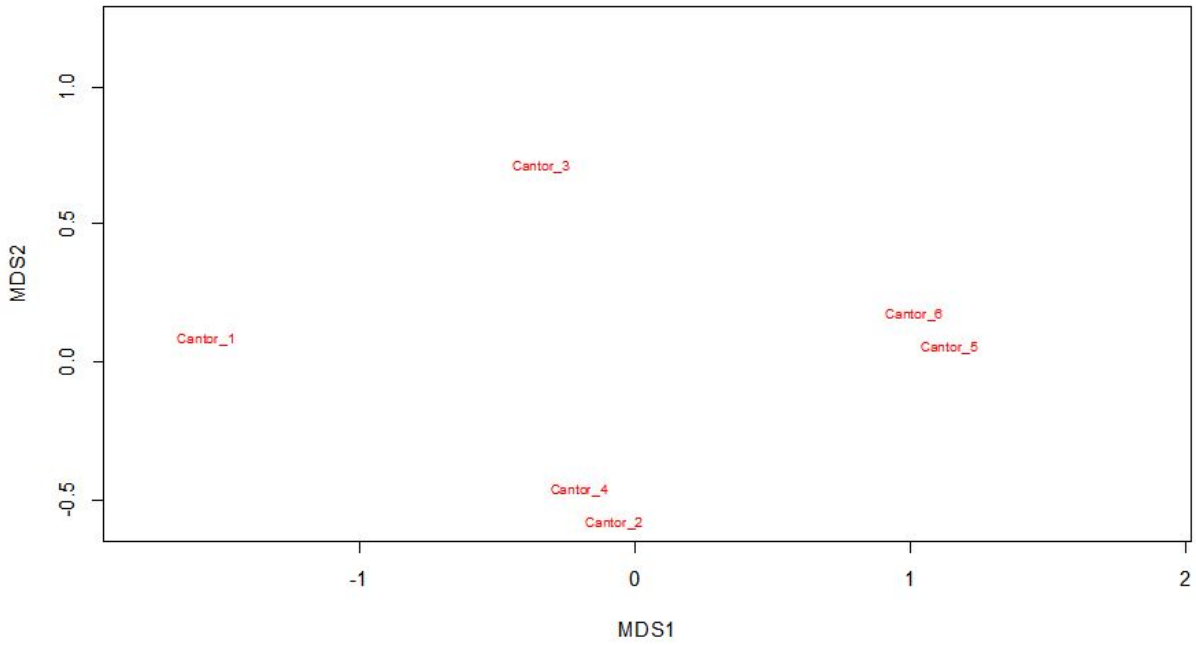
Quadro 4.4: Exemplo de código do R para escalonamento multidimensional.

| **Nº** | **Código** |
|---|---|
| 1. | library(vegan) |
| 2. | Cantor_1 <- c(1,2) |
| 3. | Cantor_2 <- c(6,7) |
| 4. | Cantor_3 <- c(5,4) |
| 5. | Cantor_4 <- c(7,5) |
| 6. | Cantor_5 <- c(9,10) |
| 7. | Cantor_6 <- c(9,9) |
| 8. | tabela <- as.data.frame( rbind(Cantor_1, Cantor_2, Cantor_3, Cantor_4, Cantor_5, Cantor_6) ) |
| 9. | tabela.dist <- dist(tabela, method = "euclidean") |
| 10. | tabela.mds <- monoMDS(tabela.dist, model = "global") |
| 11. | x <- tabela.mds$points[, 1] |
| 12. | y <- tabela.mds$points[, 2] |
| 13. | plot(tabela.mds, main = "Escalonamento Multidimensional", type = "n", xlim = range(x)+ c(0, 0.5), ylim = range(y)+ c(0, 0.5) ) |
| 14. | orditorp(tabela.mds,display="specie",col="red",air=1) |

Fonte: O autor.

A função *dist(tabela, method = "euclidean")*, no Quadro 4.4, linha 9, é uma função interna do R e calcula as distâncias entre os objetos de modo semelhante ao mostrado no Algoritmo 4.4. Essas distâncias serão processadas pela função *monoMDS()*, linha 10, e serão sugeridas as coordenadas onde os objetos deverão ser exibidos no gráfico linhas 11 e 12. As funções *plot()* e *orditorp()*, linhas 13 e 14, respectivamente, exibirão o gráfico com os objetos e, sobre o gráfico, o nomes dos objetos. Segue, na Figura 4.7, o resultado.

Figura 4.7: Escalonamento multidimensional.




Fonte: O autor.

O escalonamento multidimensional será usado neste trabalho como uma ferramenta que propicie uma percepção de semelhança entre os objetos diferentes da observada no dendrograma. Enquanto o dendrograma infere relações entre objetos por meio de hierarquias, o escalonamento multidimensional infere essas relações em um espaço onde os objetos mais semelhantes tendem a ficar visualmente próximos uns dos outros. Os cálculos da pesquisa serão feitos com base nos modelos da seção 4.1 e o escalonamento multidimensional será usado como apoio visual aos resultados obtidos nas análises de agrupamentos. Por esse motivo, os algoritmos e equações do escalonamento multidimensional não serão detalhados neste trabalho.

Capítulo Cinco

# Análise de agrupamentos linguísticos

As línguas naturais funcionam graças à interação de diferentes mecanismos, sejam eles cognitivos, motores e/ou articulatórios. Ao estudar o seu funcionamento, alguns pesquisadores optam por um aspecto específico da língua e o uso (e/ou o desenvolvimento) de uma metalinguagem formal (lógico-matemática) tem ajudado os pesquisadores nessa tarefa.

Alencar (2011) propõe a descrição de certas propriedades sintáticas das línguas através de uma língua formal. Para Russell & Norvig (2013), as línguas formais são modelos de linguagens precisamente definidos por regras (i.e., como uma sintaxe) e por significados (i.e., semântica). Russell & Norvig (2013) consideram que, ao se construir uma língua formal, desde uma língua natural, é mais plausível defini-la a partir da probabilidade de ocorrência de palavras ou letras do que a partir de um conjunto fechado de palavras. Por exemplo, a probabilidade de se encontrar na língua portuguesa a combinação de letras *qa* é menor do que *qu*.

No Capítulo 3 deste trabalho, foram apresentados alguns resultados sobre estudos referentes à frequência de letras e palavras. Os estudos referentes à identificação de línguas por meio de *n-grama* mostram percentuais de acertos que variam de 88% a 100% (e.g., Ahmed, Cha & Tappert (2004) usaram textos em língua inglesa, dinamarquesa, francesa, italiana e espanhola). Segundo os autores pesquisados, há duas variáveis que podem influenciar a identificação das línguas: (i) quantidade de dados analisada; e (ii) línguas que apresentam combinações de letras muito próximas.

A análise de agrupamentos de línguas deste trabalho considerou os seguintes critérios: usar frequência de letras ao invés de frequência de palavras, pois isso exige menos variáveis de controle (TAKCI; SOGUKPINAR, 2004)); usar um método cumulativo de frequências de *n-gramas*, pois isso gera resultados mais rápidos e assertivos na identificação de língua de um texto do que os métodos baseados em algoritmos de *Naive Bayes* e/ou ordenação de *n-grama* (AHMED; CHA; TAPPERT, 2004).

## *5.1 Metodologia aplicada na construção dos modelos propostos*

A proposta apresentada neste trabalho prevê o desenvolvimento de dois algoritmos que, doravante, serão denominados: Modelo 01 e Modelo 02. O Modelo 01 usa *bi-gramas*[1] para

[1] Conjuntos com pares de letras.

agrupar, no dendrograma, textos que compartilham uma mesma língua distinguindo-os dos agrupamentos de textos em outras línguas. O Modelo 02 será usado para verificar quantas palavras em comum existem entre textos de línguas que aparecerem muito próximas no dendrograma e, com isso, sugerir uma relação entre as línguas desses textos considerando a variação interna das línguas estudadas[2].

Para os Modelos propostos abaixo, as seguintes etapas de tratamento de dados foram feitas: (i) transformar todos os caracteres dos textos em letras minúsculas; (ii) eliminar dos textos todos os caracteres especiais (e.g., @, $, &, etc); e (iii) eliminar dos textos números (e.g., 1,2,4,7, etc.). Os textos selecionados para as análises foram escritos com letras do alfabeto europeu e latino e gravados em arquivos nos formato *.txt* com tipo de codificação binária UTF-8 [3]; isso para garantir a uniformidade de caracteres padrões e evitar conflitos ao trabalhar com arquivos com tipos de codificação diferentes. Não foram analisadas línguas que usam ideogramas como o chinês ou o japonês, por exemplo.

O Modelo 01 baseia-se na teoria de *n-gramas* e foi desenvolvido com o propósito de agrupar textos que compartilham uma mesma língua. Pretende-se com esse modelo identificar as línguas que são tão próximas entre si que os agrupamentos não conseguem distinguí-las. O Modelo 02 será usado para analisar as línguas que, no Modelo 01, apareceram muito próximas ou num mesmo agrupamento. Pretende-se, comparando os conjuntos de palavras dessas línguas, verificar suas proximidades em termos de variedades de uma língua (dialetos).

### *5.1.1 Modelo 01*

O principal objetivo do Modelo 01 é processar um conjunto de textos em diferentes línguas e construir um dendrograma agrupando os textos de mesma língua em grupos específicos. Os objetivos secundários do Modelo 01 são: (i) exibir um gráfico em escalonamento multidimensional para visualizar o quão próximos podem os agrupamentos estar uns dos outros; (ii) exibir *heatmaps* [4] das matrizes de *bi-gramas* para verificar visualmente se existem distinções entre matrizes de *bi-gramas* de diferentes línguas.

O Algoritmo 5.1 explica como o dendrograma é construído.

[2] Ver discussão sobre variação linguística no Capítulo 2, Seção 2.4 Língua e dialeto

[3] O UTF-8 é um tipo de codificação binária usada para representar caracteres em diferentes línguas.

[4] Trata-se de uma matriz numérica que, ao invés de exibir números nas intersecções entre linhas e colunas, exibe escalas de cores, podendo, por exemplo, exibir cores frias para os valores mais baixos da matriz e cores quentes para os valores mais altos.

**Algoritmo 5.1** Exibir o dendrograma com os agrupamentos de texto de mesma língua.

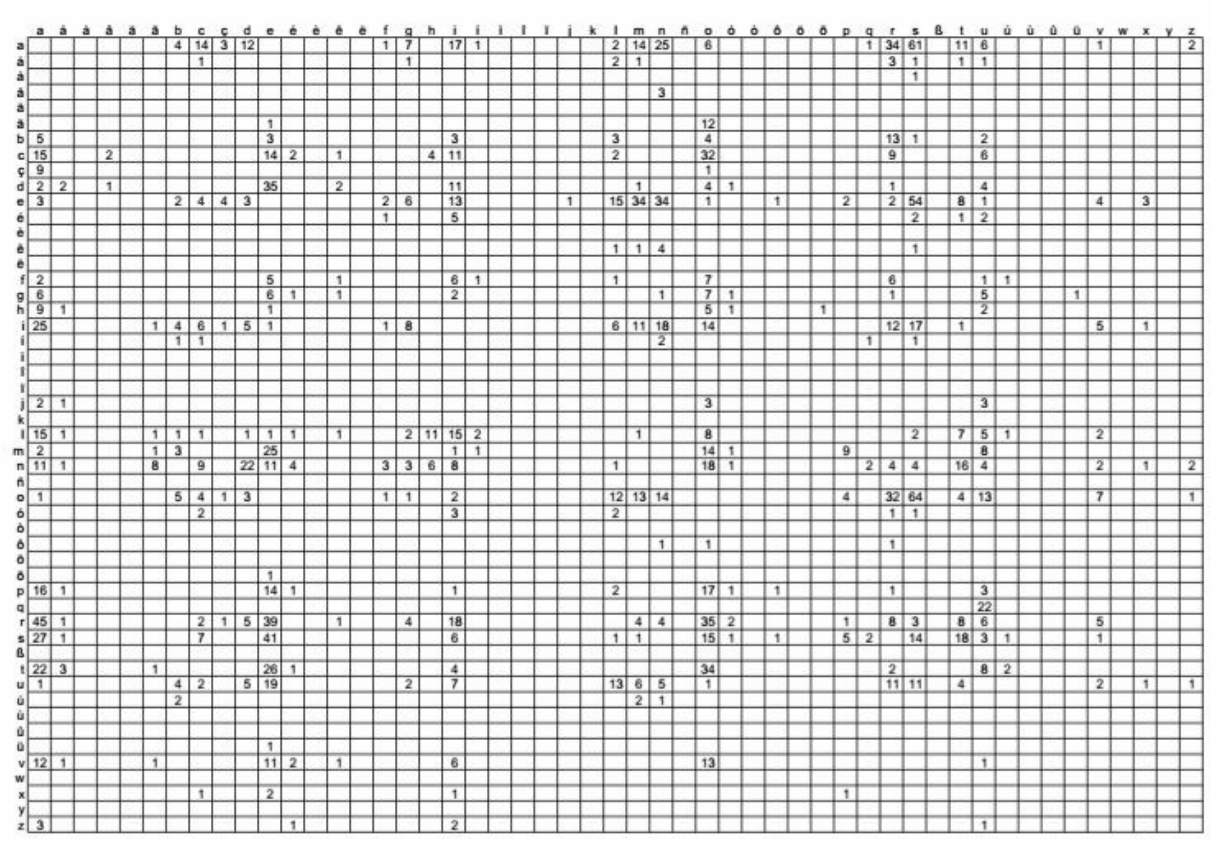
1: **Parâmetro de entrada:**
2: - Endereço de onde estão os arquivos em formato .txt;
3: - Nome do método de cálculo de distância (e.g, *Distância euclidiana*);
4: - Nome do método de cálculo de distância intragrupo (e.g., *Distância média*);
5: - Texto (Sim/Não) exibir o gráfico escalonamento multidimensional;
6: - Texto (Sim/Não) exibir os *heatmaps* dos *bi-gramas* dos textos;
7: **Parâmetro de saída:**
8: - Dendrograma com os agrupamentos de textos de mesma língua;
9: - Gráfico do escalonamento multidimensional;
10: - *Heatmaps* dos arquivos carregados;
11: **Início:**
12: *Passo 1:* **Enquanto** houver arquivos para ler no diretório especificado **faça**
13: *Passo 1.1:* Armazenar em uma lista **A** o texto;
14: *Passo 1.2:* Remover do texto em **A** os caracteres especiais, números e excesso de tabulações;
15: *Passo 1.3:* Trocar todos os caracteres do texto em **A** para letras minúsculas;
16: *Passo 1.4:* Armazenar em uma lista **L** um exemplar de cada caractere lido;
17: *Passo 2:* Criar uma lista **M** para armazenar matrizes (**L** por **L** - uma matriz para cada texto em **A**);
18: *Passo 3:* **Enquanto** houver textos na lista **A faça**
19: *Passo 3.1:* Ler cada par de letras na lista **A** da vez, identificar a intersecção desses pares na matriz **M** da vez e somar 1.
20: *Passo 4:* Criar uma Matriz **T** e armazenar em cada linha de **T** um vetor com cada matriz **M** (e.g., Figura 5.1);
21: *Passo 5:* Construir e exibir, a partir da matriz **T**, um dendrograma com os métodos de distância entre objeto e distância intragrupo informados inicialmente;
22: *Passo 6:* Se foram solicitados inicialmente com *Sim*, exibir o gráfico com o escalonamento multidimensional e/ou os *heatmaps* de *bi-gramas* de textos;

Os algoritmos apresentados neste Capítulo estão transcritos no Apêndice C deste trabalho, em linguagem R, e serão estruturados aqui em forma de passos. O intuito é simplificar a apresentação dos Modelos 01 e 02 generalizando funções e processos.

O Algoritmo 5.1 criará duas listas: a lista **A** armazena em cada linha um texto pré-processado do diretório de arquivos; e a lista **M** contém uma matriz (e.g. Figura 5.1) na qual as linhas e colunas representam as letras identificadas por **L** (acentuadas ou não) nos textos lidos. Para cada texto existirá uma matriz **M**. Os passos entre as linhas 9 e 11 do Algoritmo 5.1 fazem um pré-processamento nos textos eliminando os caracteres que não serão analisados, como números e caracteres especiais, e transforma todas as letras dos textos em minúsculas. O objetivo é reduzir o tamanho do corpus evitando processar dados irrelevantes e, ao transformar as letras para minúscula, evitar a duplicidade de

combinações de mesmos pares de letras (e.g., "Oi"e "oi"representam a mesma palavra, mas para linguagens de programação *case-sensitive* [5] são combinações diferentes) .

Tabela 5.1: Exemplo de Matriz de custo para pares de letras.

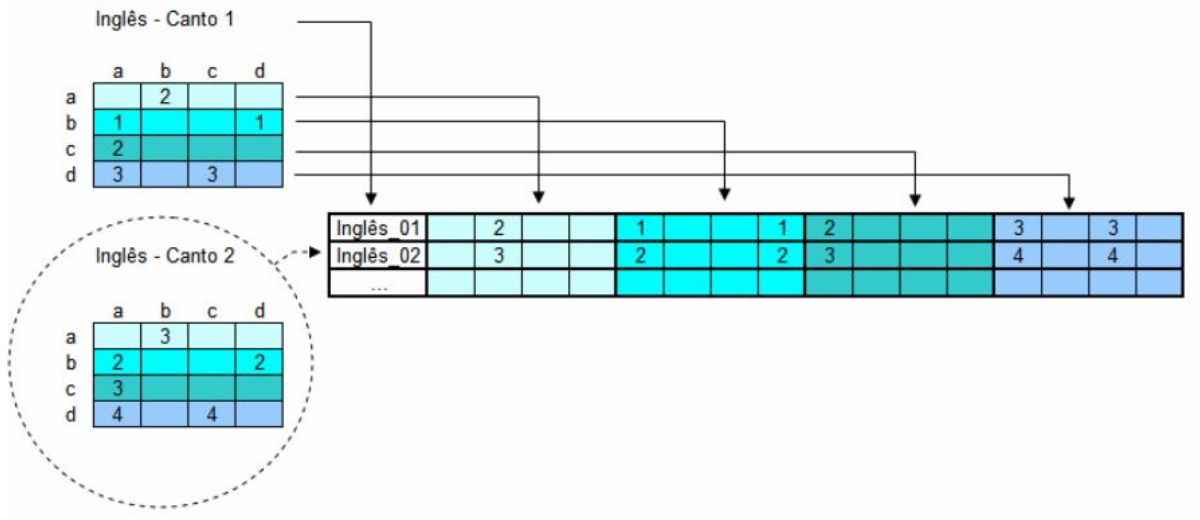


Fonte: O autor.

O passo descrito na linha 18, Algoritmo 5.1, constrói a matriz **T** transformando cada matriz de **M** em um vetor e, assim, consolidando a matriz **T**. No exemplo da Figura 5.1, dois textos (os textos chamados de Canto 1 e Canto 2 escritos em inglês) foram transformados em duas matrizes de custo, cada matriz representando um desses textos e cada linha de cada matriz de custo é desmembrada sequencialmente para se tornar uma linha em **T**. A matriz **T** é uma tabela similar à Tabela 4.1; a primeira coluna armazena os nomes dos objetos, no caso o nome de um texto/língua; e as demais colunas são os atributos desses objetos.

[5] São linguagem de programação que diferenciam nomes em caracteres maiúsculos de minúsculos, por exemplo, as variáveis *nome01* e *Nome01* tem o mesmo nome, mas são duas variáveis diferentes.

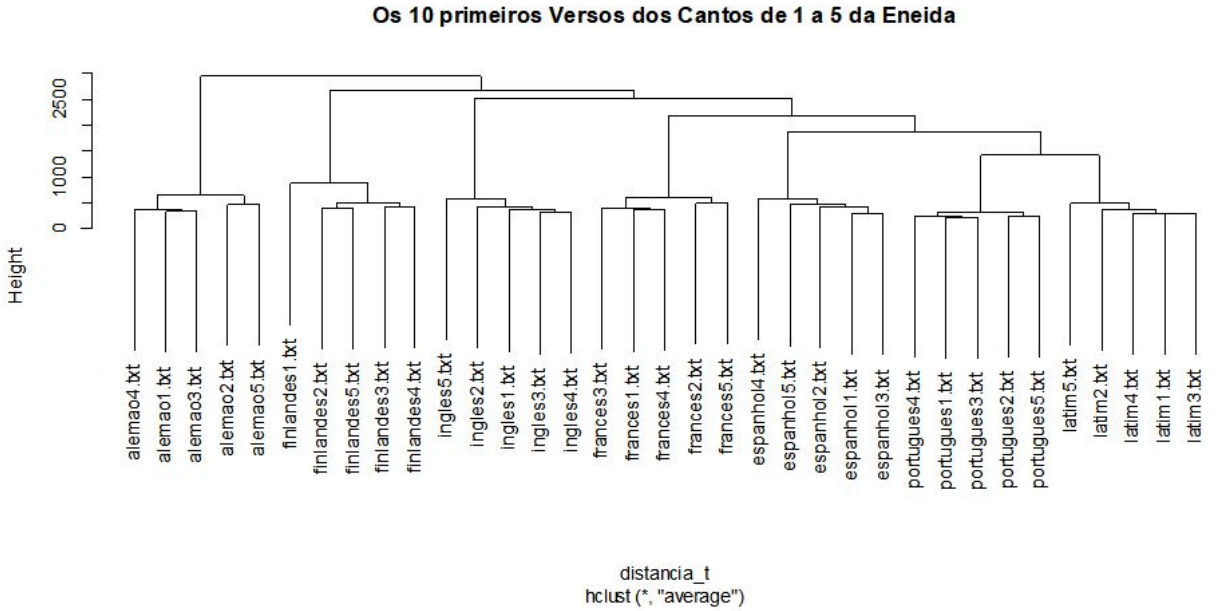
Figura 5.1: Exemplo de consolidação das matrizes dos textos em uma tabela de objetos.




Fonte: O autor.

A matriz **T** será usada para construção do dendrograma. Por exemplo, em um diretório estão armazenados os cinco primeiros Cantos do poema épico *Eneida* do autor *Virgílio*; cada Canto estará em um arquivo e, como são sete traduções desse poema, existirão 35 arquivos nessa pasta; ao carregar o Modelo 01 com os parâmetros: o endereço do diretório em questão; *distância euclidiana*; *distância média*; *sim* para o escalonamento multidimensional; e *sim* para os *heatmaps*, o dendrograma será construído conforme a Figura 5.2.

Figura 5.2: Exemplo de dendrograma do Modelo 1.

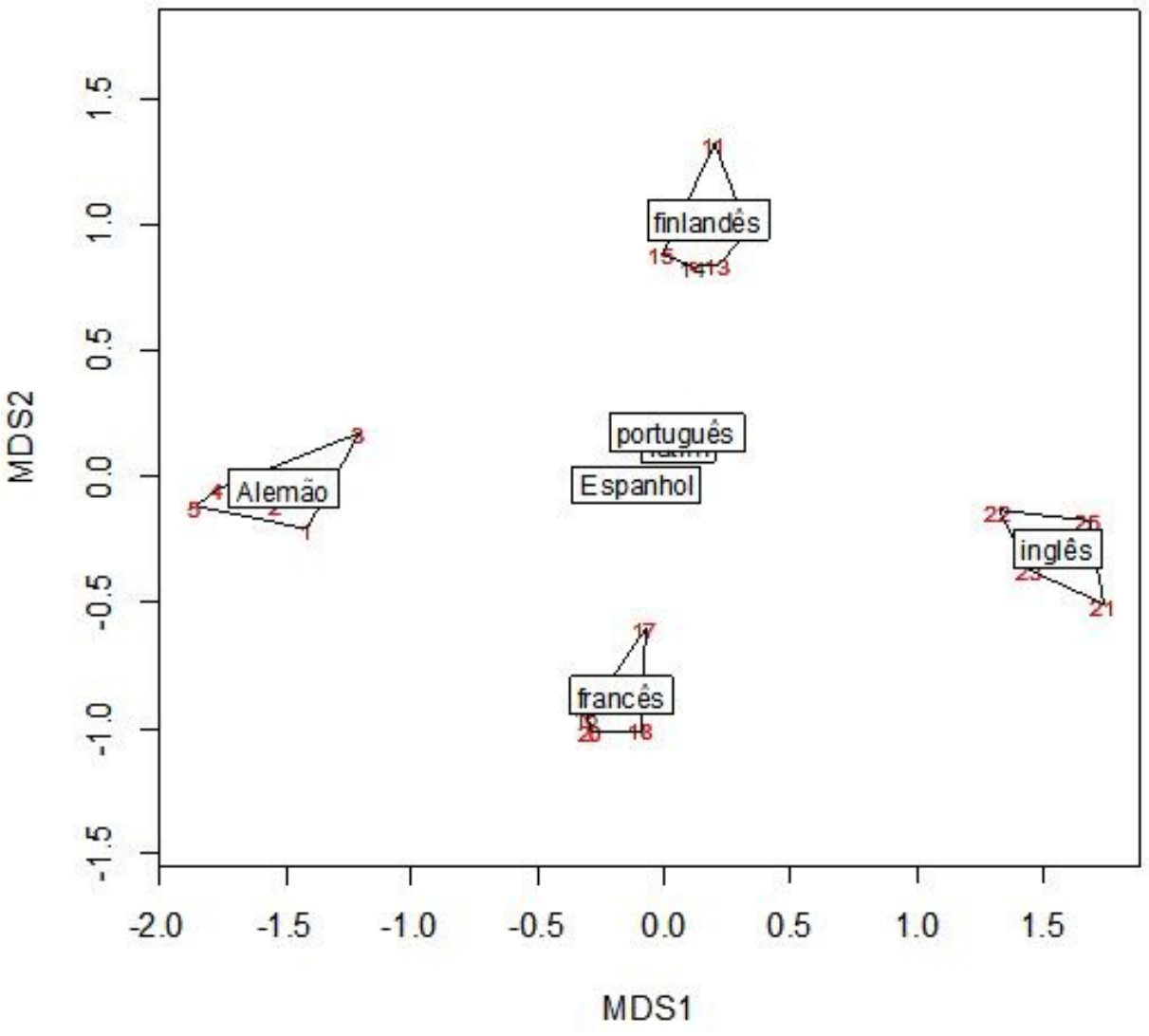



Fonte: O autor.

O dendrograma da Figura 5.2 reuniu, em sete grupos com cinco elementos cada, todos os textos que compartilham uma mesma língua, ou seja, na análise de agrupamento a matriz **T** apresenta atributos que são relevantes para distinguir textos de línguas diferentes e relevantes para agrupar os textos de mesma língua.

Visualmente, a semelhança entre os objetos (i.e., textos) e os grupos ao qual pertencem cada objeto pode ser percebida no gráfico de escalonamento multidimensional (Figura 5.3).

Figura 5.3: Exemplo de Escalonamento Multidimensional do Modelo 1.



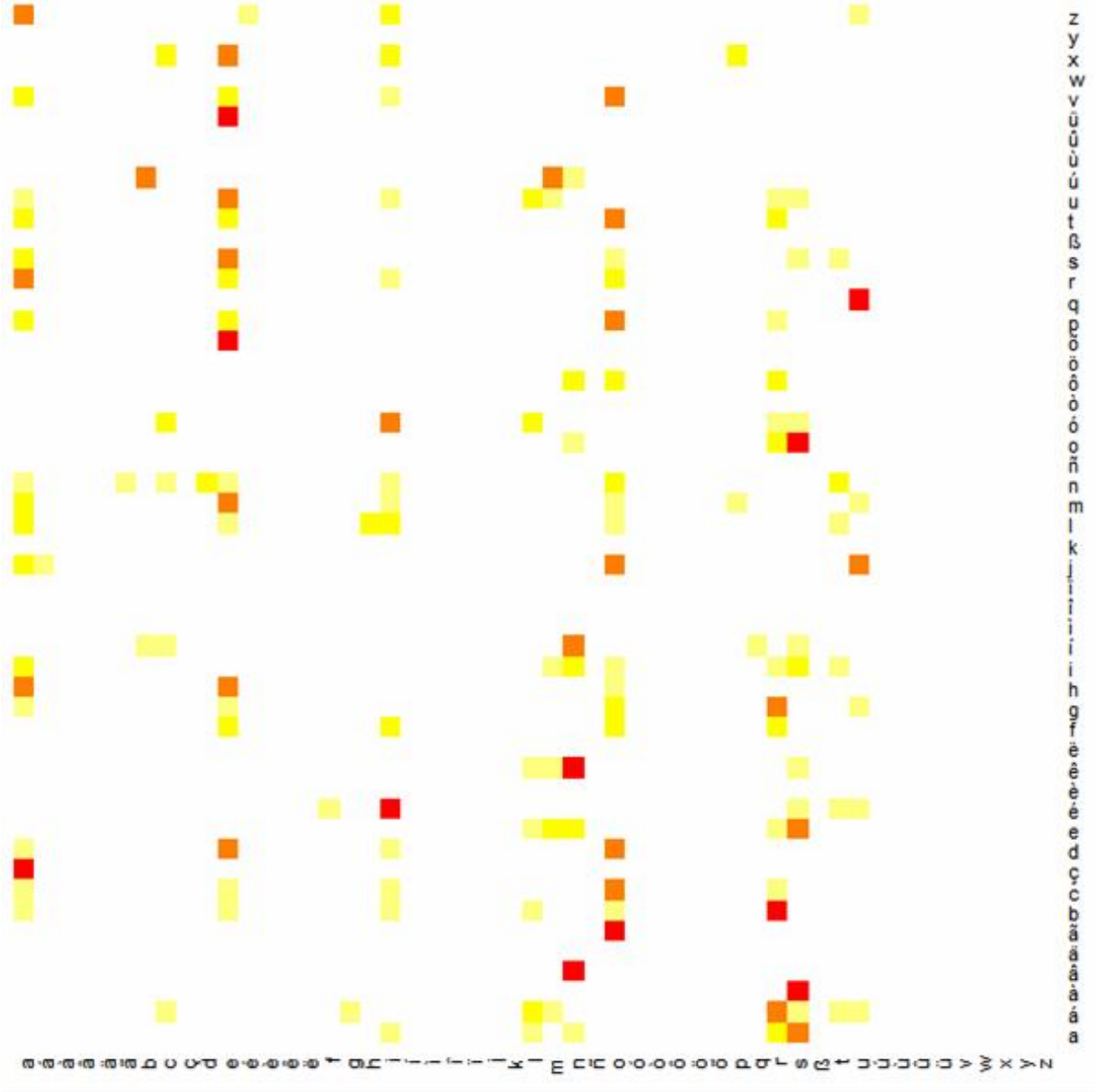

Fonte: O autor.

Na Figura 5.3, cada texto é representado por um número; os grupos que eles formam estão cercados por linhas e são representados por rótulos (e.g., alemão, espanhol, etc). A distância entre os grupos sugere semelhança entre eles quando estão muito próximos e dissemelhança quando estão mais distantes. No exemplo da Figura 5.3 a semelhança

entre o espanhol, português e latim são maiores, por isso estão tão próximos ao ponto do rótulo do português sobrepor o do latim.

Outra forma de notar visualmente as diferenças no uso de pares de letras entre as línguas é por meio dos *heatmaps*. Na Figura 5.4, os tons de amarelo representam os valores próximos a zero; os tons em laranja representa valores intermediários; e os tons em vermelho representam os valores mais altos encontrados.

Figura 5.4: Exemplo de heatmap do R.



Fonte: O autor.

### *5.1.2 Modelo 02*

Para o Modelo 2, a proposta é verificar quantas palavras idênticas existem entre conjuntos de textos de diferentes línguas. Pelo Modelo 02 serão analisadas as línguas que não conseguiram construir, no dendrograma, agrupamentos de texto com uma única língua, ou seja, agrupamentos mistos com texto de mais de uma língua. O objetivo é verificar, por meio das palavras comuns a essas línguas, o quão próximas as línguas estão em termos de léxico em comum. Quanto mais palavras em comum houver entre os textos, mais provável será tratar-se de variedades de uma línguas (ou línguas muito próximas).

O Algoritmo 5.2 recebe como parâmetros de entrada o nome de um diretório onde estão os textos (o diretório deve conter um texto para cada língua) e retorna o diagrama de *Venn*[6] mostrando quantas palavras em comum existem entre as línguas dos textos analisados.

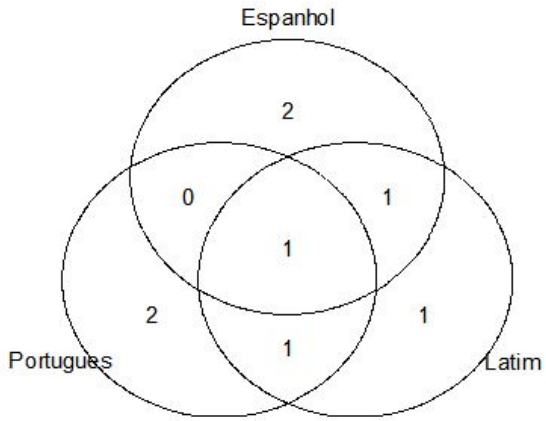
**Algoritmo 5.2** Exibir o diagrama de *Venn* com as quantidades de palavras em comum existentes nos textos de diferentes línguas.

1: **Parâmetro de entrada:**
2: - Endereço de onde estão os arquivos em formato .txt;
3: **Parâmetro de saída:**
4: - Diagrama de *Venn*;
5: **Início:**
6: *Passo 1:* **Enquanto** houver arquivos para ler no diretório especificado **faça**
7: *Passo 1.1:* Armazenar em uma lista **A** todas as palavras do texto;
8: *Passo 2:* Contar quantas palavras em comum existem entre os membro da lista **A**;
9: *Passo 3:* Gerar o diagrama de *Venn*;

Por exemplo, ao passar como parâmetro de entrada do Algoritmo 5.2 o endereço de um diretório que contem três arquivos (um em português com as palavras "pura", "vinho", "deserto"e "cachorro"; um em latim com as palavras "pura", "vino", "deserto"e "canem"; e um em espanhol com as palavras "pura", "vino", "desierto"e "perro"), o algoritmo retornará o diagrama de *Venn* conforme a Figura 5.5.

[6]Representa graficamente conjuntos por meio de círculos; dentro de cada círculo são exibidos números que representam quantos elementos o círculo (ou conjunto) tem e as sobreposições desses círculos exibem valores, ou quantidades de elementos, que são compartilhados por esses círculos sobrepostos.

Figura 5.5: Diagrama de *Venn*.




Fonte: O autor.

Por meio do diagrama de *Veen*, Figura 5.5, verifica-se que existe uma palavra que é comum aos três textos; uma palavra em comum entre o português e o latim; nenhuma entre o português e o espanhol; etc.

## 5.2 Aplicação dos modelos e resultados

Nesta primeira parte da seção pretende-se: (i) verificar qual dos cálculos de distância entre objetos e dos cálculos de distância intergrupos geram, na análise de agrupamentos, resultados próximos à realidade de grupos já conhecidos; (ii) verificar se os *bi-gramas* têm propriedades suficientes para agrupar ou distinguir textos de línguas diferentes no dendrograma, no escalonamento multidimensional e nos *heatmaps*; (iii) verificar qual é a quantidade mínima de palavras necessárias para que seja possível identificar a formação de agrupamentos de mesma língua no dendrograma; (iv) validar, por meio de índices baseados em critérios interno e externo, se os resultados do dendrograma estão de acordo com o padrão de agrupamento esperado para os dados usados. Essas informações serão úteis para reduzir o tempo de processamento dos dados, conhecer qual é a quantidade mínima para que a análise funcione e validar o funcionamento do Modelo 01.

O primeiro conjunto de textos usado nesta pesquisa continha sete traduções (alemão, espanhol, inglês, finlandês, francês, latim e português) dos poemas épicos de *Virgilio* chamado *Eneida*. Cada um dos cinco primeiros Capítulos, ou Cantos, desse livro foi gravado em um arquivo texto (formato .txt) com o nome da língua e o número do Canto selecionado. Depois de aplicar esse conjunto de textos a diferentes configurações de agrupamentos no Algoritmo 5.1, (Modelo 01) obtiveram-se os seguintes resultados.

Tabela 5.2: Teste de configurações das análises de agrupamento

| Distância entre Objetos | Distância intergrupos | Agrupou corretamente? | Índices Internos | Índices |
|---|---|---|---|---|
| Euclidiana | mínima | sim | *Dunn* | 1,045 |
| | máxima | sim | *Davies-Bouldin* | 0,824 |
| | média | sim | *Silhouette* | 0,751 |
| Manhattan | mínima | sim | *Dunn* | 1,032 |
| | máxima | sim | *Davies-Bouldin* | 0,781 |
| | média | sim | *Silhouette* | 0,734 |
| Canberra | mínima | sim | *Dunn* | 1,032 |
| | máxima | sim | *Davies-Bouldin* | 0,781 |
| | média | sim | *Silhouette* | 0,542 |

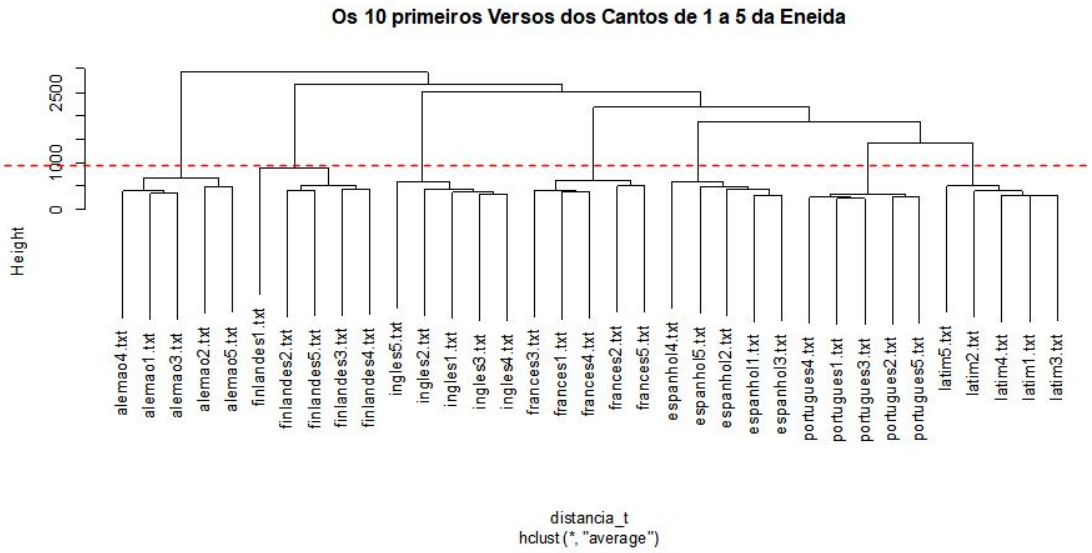
Fonte: O autor.

Os textos das traduções da *Eneida* foram submetidos a todas as configurações na Tabela 5.2. Os dendrogramas agruparam todos os 35 arquivos em 7 grupos, cada grupo com textos de uma língua específica. Como todas as combinações de distâncias conseguiram identificar os agrupamentos conforme o esperado, não foi necessário calcular os índices baseados em critérios externos porque os grupos identificados pelos dendrogramas são os mesmos grupos formados pelos arquivos quando agrupados por tradução.

Em relação aos índices baseados em critérios internos, o cálculo de distâncias euclidianas foi o que obteve os melhores resultados para o índice de *Silhouette* com 0,751 (quanto mais próximo de 1 melhor é a formação de agrupamentos), e o índice de *Dunn* com 1,045 (quanto mais distante de zero melhor é a formação de agrupamentos). A *distância média intergrupos* distinguiu um pouco melhor os agrupamentos e, por essa razão, optou-se por usar, nesta pesquisa, a distância euclidiana e a distância média intergrupo.

A Figura 5.6 exibe o dendrograma baseado na distância euclidiana e na *distância média* intergrupos.

Figura 5.6: Dendrograma com os 5 primeiros cantos da Eneida.




Fonte: O autor.

A linha tracejada em vermelho na Figura 5.6 demarca, nas intersecções com as linhas do dendrograma, os pontos onde é possível identificar sete agrupamentos. Cada agrupamento abaixo da linha tracejada contém textos de uma mesma língua; as distância entre os textos estão na escala de valores ao lado esquerdo do dendrograma e, quanto menor a distância entre os textos, mais atributos semelhantes eles compartilham. As relações entre os grupos aparecem acima da linha tracejada e, quanto maior a distância entre os agrupamentos, menos atributos semelhantes eles compartilham. Por exemplo, o agrupamento com textos em latim é mais semelhante ao agrupamento com textos em português e menos semelhante ao agrupamento com textos em alemão.

As diferenças entre os atributos, no caso *bi-gramas*, das línguas podem ser percebidas visualmente na Figura 5.7.

Figura 5.7: *Heatmaps* do Modelo 1 para os 5 cantos do livro Eneida em sete traduções.

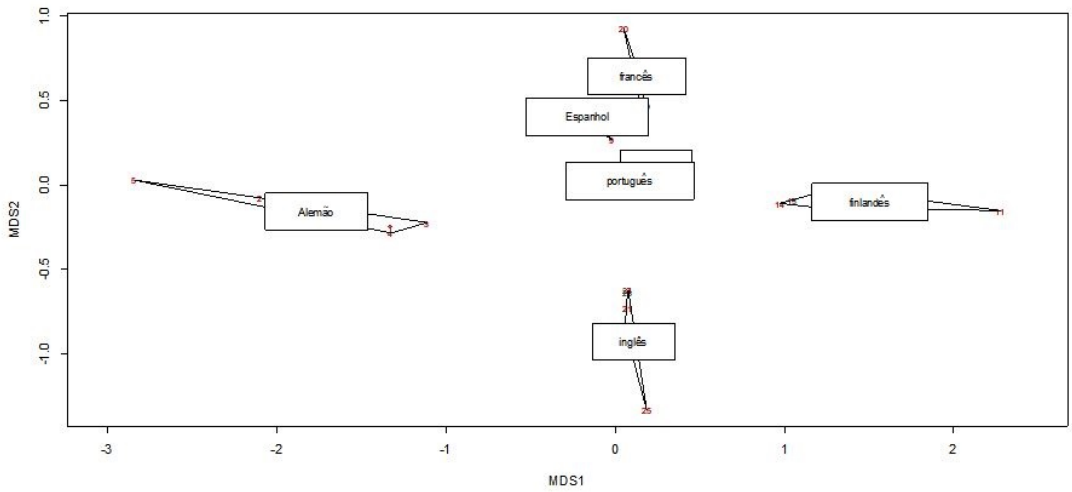

Fonte: O autor.

É possível verificar visualmente na Figura 5.7[7] um padrão de cores entre as matrizes de mesma língua e que esse padrão difere para as línguas. Preliminarmente, é possível verificar que existem diferenças no modo como as línguas organizam suas letras.

[7]No Apêndice B existem exemplos em maior escala.

Na perspectiva do Escalonamento Multidimensional, os agrupamentos dessas línguas aparecem conforme a representação na Figura 5.8.

Figura 5.8: Escalonamento multidimensional com os 5 primeiros cantos da Eneida.




Fonte: O autor.

Na Figura 5.8 os textos em latim e em português aparecem muito próximos e, por esse motivo, o rótulo do português sobrepôs o rótulo do latim.

No Quadro 5.1, seguem os resultados dos índices baseados em critérios internos e externos.

Quadro 5.1: Resultados dos índices baseados em critérios interno e externo para 5 primeiros Cantos da Eneida (aprox. 6.000 palavras por arquivo).

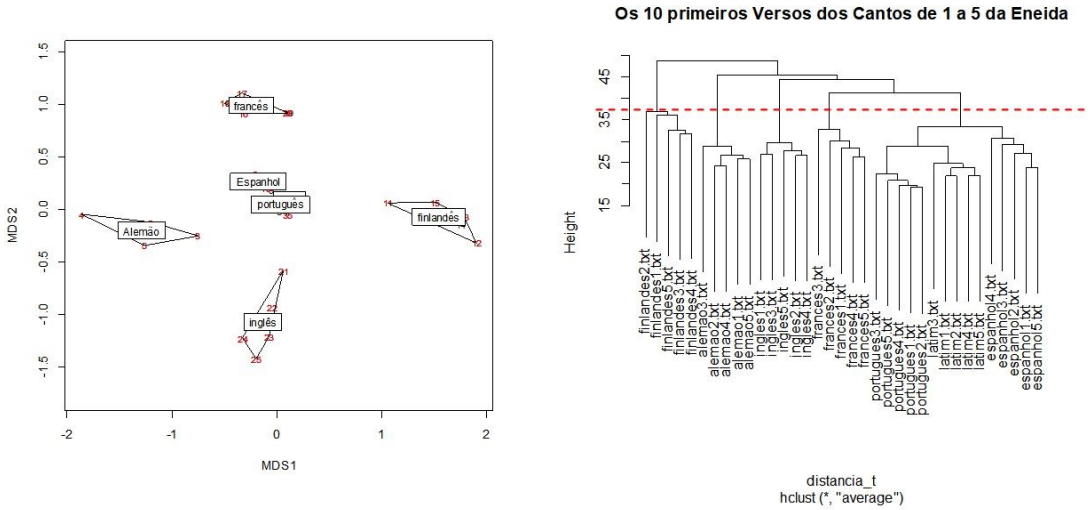
| **Índices** | **Resultados** | **Descrição** |
|---|---|---|
| *Rand* | 1 | Para esses três índices o resultado igual a 1 indica que existe similaridade entre as partições e grupos. |
| *Jaccard* | 1 | |
| *Fowlkes & Mallows* | 1 | |
| *Dunn* | 1,045 | Essa é a primeira medida feita pelo índice de *Dunn* neste trabalho. Como o índice considera que valores altos indicam boas formações de grupos, esse resultado será usado como parâmetro para as próximas comparações. |
| *Davies-Bouldin* | 0,824 | O índice de *Davies-Bouldin* considera que valores próximos a zero indicam boas formações de agrupamentos; como essa é a primeira vez que o índice foi usado, esse resultado será considerado nas próximas comparações. |
| *Silhouette* | 0,751 | Os valores desse índice variam de -1 a +1 e, quanto mais próximo de +1, mais bem formados são os agrupamentos. |

Fonte: o autor.

Os arquivos analisados tinham em média 6.000 palavras. Em um novo teste, reduzindo

a quantidade média para 100 palavras por arquivo, o dendrograma e o escalonamento multidimensional ficaram conforme a Figura 5.9.

Figura 5.9: Dendrograma e escalonamento multidimensional com média de 100 palavras por arquivo.




Fonte: O autor.

No dendrograma da Figura 5.9, os textos em português, latim e espanhol ficaram em um mesmo agrupamento. Não existe nesse dendrograma um ponto onde a linha tracejada cruze horizontalmente as hierarquias e localize as intersecções onde os agrupamentos são formados exclusivamente por textos de mesma língua. No Quadro 5.2 são apresentados os índices baseados em critérios interno e externo coletados.

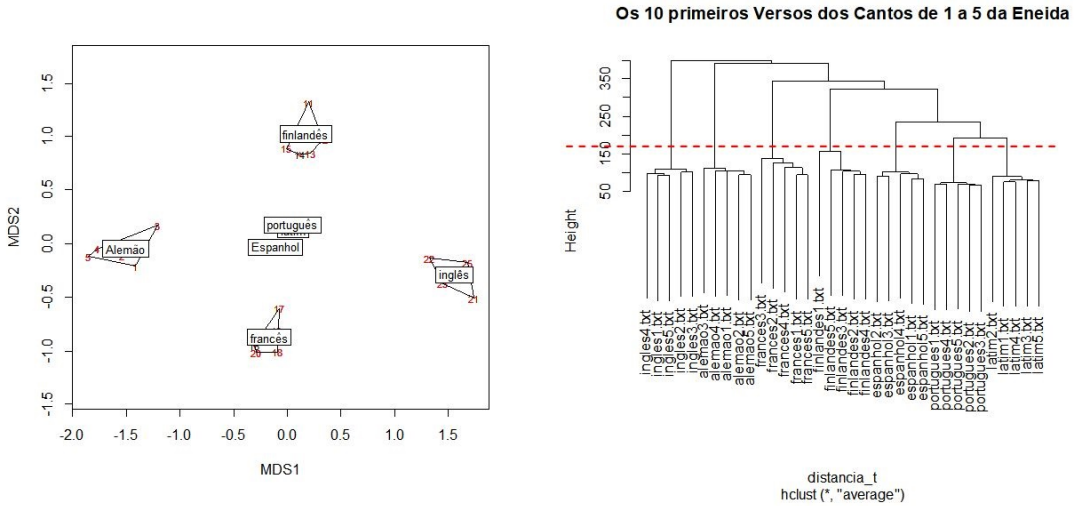
Quadro 5.2: Resultados dos índices baseados em critérios interno e externo para os 5 primeiros Cantos da Eneida (aprox. 100 palavras por arquivo).

| **Índices** | **Resultados** | **Descrição** |
|---|---|---|
| *Rand* | 0,862 | Para esses índices, os valores próximos de 1 indicam que os agrupamentos esperados e os calculados são semelhantes. No entanto, comparando esses índices com os índices do Quadro 5.1, percebe-se que houve uma queda; antes os índices eram 1 (indicando que os agrupamentos conhecidos e os calculados eram iguais) e agora estão abaixo de 1 (indicando que não são mais iguais). |
| *Jaccard* | 0,434 | |
| *Fowlkes & Mallows* | 0,640 | |
| *Dunn* | 0,853 | Comparado com o índice do Quadro 5.1, esse resultado foi menor; quanto mais próximo de zero for o índice de *Dunn*, menos semelhantes são os agrupamento conhecidos dos objetos e os agruapementos calculados. |
| *Davies-Bouldin* | 1,665 | Em comparação ao resultado do Quadro 5.1, esse índice aumentou e isso indica que não houve melhora na formação dos agrupamentos. |
| *Silhouette* | 0,277 | O valor diminuiu se comparado ao Quadro 5.1 e isso indica que a formação de agrupamentos calculada é pouco semelhante ao agrupamento conhecido dos dados. |

Fonte: o autor.

Uma média de 100 palavras por arquivo não é suficiente para que o dendrograma construa agrupamentos com textos de mesma língua. O mesmo teste foi feito com médias de 500, 750 e 1.000 palavras por texto. A média de 1.000 palavras por texto gerou resultados satisfatórios no agrupamento de textos de mesma língua e índices baseados em critérios interno e externo próximos ao do Quadro 5.1. Com uma quantidade média de 1.000 palavras por arquivo, o dendrograma e o escalonamento multidimensional ficaram conforme a representação na Figura 5.10.

Figura 5.10: Dendrograma e escalonamento multidimensional com média de 1.000 palavras por arquivo.




Fonte: O autor.

No Quadro 5.3, seguem os índices baseados em critérios interno e externo para uma média de 1.000 palavras por arquivo.

Quadro 5.3: Resultados dos índices baseados em critérios interno e externo para os 5 primeiros Cantos da Eneida (aprox. 1.000 palavras por arquivo)

| **Índices baseados em critérios externos** | | **Índices baseados em critérios internos** | |
|---|---|---|---|
| *Rand* | 1 | *Dunn* | 1.081 |
| *Jaccard* | 1 | *Davies-Bouldin* | 0.900 |
| *Fowlkes & Mallows* | 1 | *Silhouette* | 1 |

Fonte: o autor.

A média de 1.000 palavras por arquivo mostrou-se uma boa medida para que o dendrograma reúna os textos em grupos que compartilham uma mesma língua. Além disso, os índices baseados em critérios interno e externo apontam que há boa identificação de grupos. No escalonamento multidimensional, apesar de os agrupamentos das línguas portuguesa, espanhola e latina terem ficado próximos, é possível identificá-los. A partir de agora esta pesquisa adotará os seguintes padrões para a análise de agrupamentos: o cálculo de distância entre objetos será euclidiano; o cálculo de distância entre grupos será a *distância média*; e os arquivos textos terão em média 1.000 palavras.

O próximo conjunto de textos a ser analisado contém 30 traduções da Bíblia (Novo testamento). Cada uma dessas traduções foi dividida em 05 arquivos com partes diferentes da bíblia e com média de 1.000 palavras por arquivo. A partir desse processo, foram criados 150 arquivos e no Quadro 5.4 estão os nomes das línguas das Bíblias consultadas.

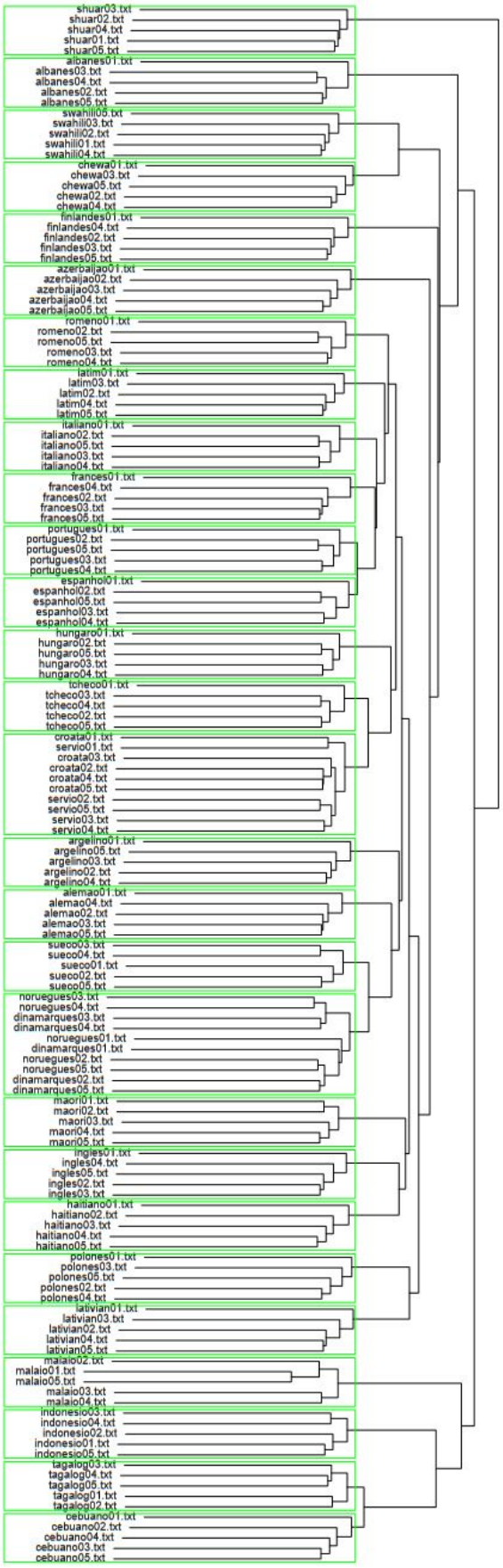
Quadro 5.4: Lista com o nome das línguas usadas na tradução da Bíblia.

| **Nº** | **Pais** | **Língua** |
|---|---|---|
| 1 | Azerbaijão | Azeri |
| 2 | Letônia | letã |
| 3 | Polônia | Polaca |
| 4 | Romênia | Romena |
| 5 | Argélia | Árabe |
| 6 | Suécia | Sueca |
| 7 | Dinamarca | Dinamarquesa |
| 8 | Noruega | Norueguesa |
| 9 | França | Francesa |
| 10 | Brasil | Portuguesa |
| 11 | Espanha | Espanhola |
| 12 | Itália | Latim |
| 13 | Itália | Italiano |
| 14 | Finlândia | Finlandesa |
| 15 | Checoslováquia | Checa |
| 16 | Sérvia | Sérvia |
| 17 | Croácia | Croata |
| 18 | Hungria | Húngara |
| 19 | Albânia | Albanesa |
| 20 | Alemanha | Alemã |
| 21 | Inglaterra | Inglesa |
| 22 | Indonésia | Indonésia |
| 23 | Filipinas | Tagalog |
| 24 | Filipinas | Cebuano |
| 25 | Nova Zelândia | Maori |
| 26 | Haiti | Haitiana |
| 27 | Quênia | Swahili |
| 28 | Malaui | Chewa |
| 29 | Malásia | Malaia |
| 30 | Peru | Shuar |

Fonte: O autor.

A Figura 5.11 exibe o dendrograma construído a partir dos textos da Bíblia. Devido à grande quantidade de textos, na imagem não é possível ler o nome de cada um, por esse motivo optou-se por segmentar a imagem e analisá-la separadamente.

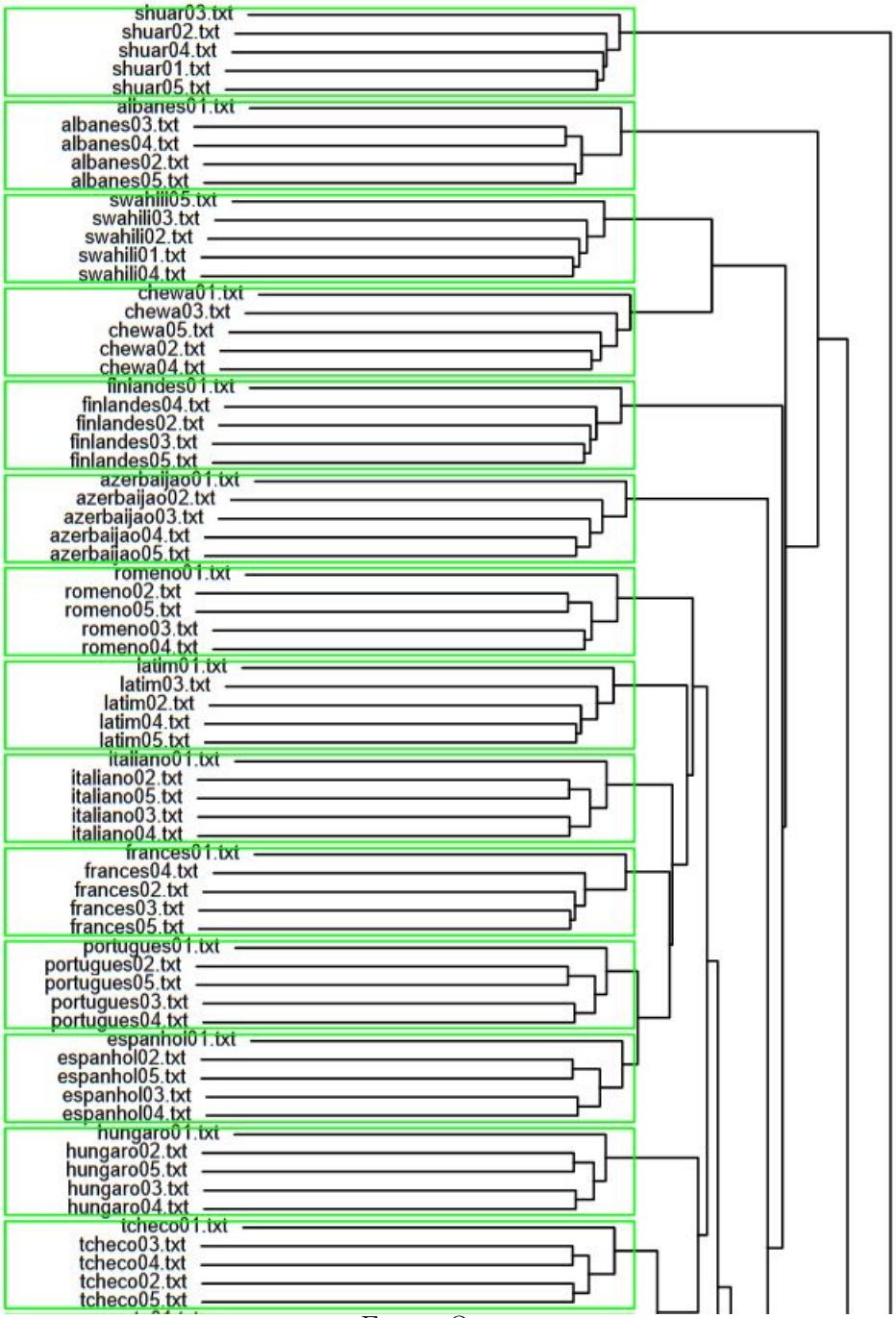
Figura 5.11: Dendrograma das 30 traduções da bíblia.




Fonte: O autor.

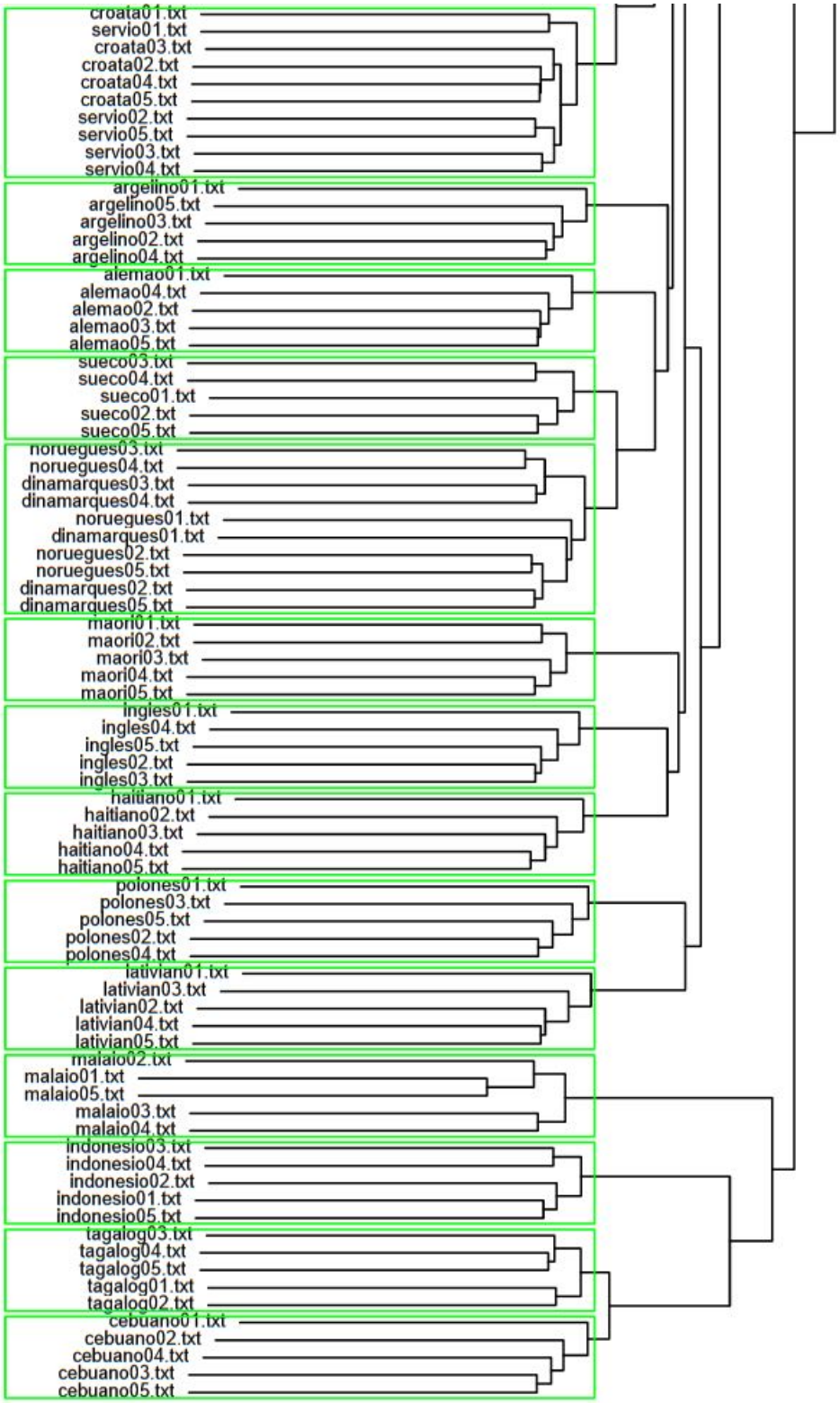
O dendrograma da Figura 5.12 exibe os primeiros 14 agrupamentos de textos da Figura 5.11. Todos esses agrupamentos são homogêneos, ou seja, cada um deles só contém textos que compartilham uma mesma língua.

Figura 5.12: Dendrograma das 30 traduções da bíblia - Parte 01




Fonte: O autor.

Figura 5.13: Dendrograma das 30 traduções da bíblia - Parte 02




Fonte: O autor.

Na Figura 5.13 há dois agrupamentos que reúnem textos de diferentes língua: o agrupamento onde aparecem as línguas sérvia e croata e o agrupamento onde aparecem as línguas norueguesa e dinamarquesa. Os demais agrupamentos reúnem textos de uma

mesma língua.

Na análise de agrupamentos, cada agrupamento reúne um conjunto de objetos que compartilham entre si atributos semelhantes. Para o dendrograma da Figura 5.13, esses atributos são os *bi-gramas*; dessa forma, os textos nos agrupamentos formados pelas língua dinamarquesa/norueguesa e sérvia/croata têm *bi-gramas* que são comuns entre elas e sugerem que essas línguas são semelhantes (com base nesse critério de *bi-gramas*).

No Quadro 5.5, seguem os resultados dos índices baseados em critérios internos e externos.

Quadro 5.5: Resultados dos índices das 30 traduções da bíblia.

| **Índices baseados em critérios externos** | | **Índices baseados em critérios internos** | |
|---|---|---|---|
| *Rand* | 0.994 | *Dunn* | 0.661 |
| *Jaccard* | 0.834 | *Davies-Bouldin* | 1.516 |
| *Fowlkes & Mallows* | 0.911 | *Silhouette* | 0.476 |

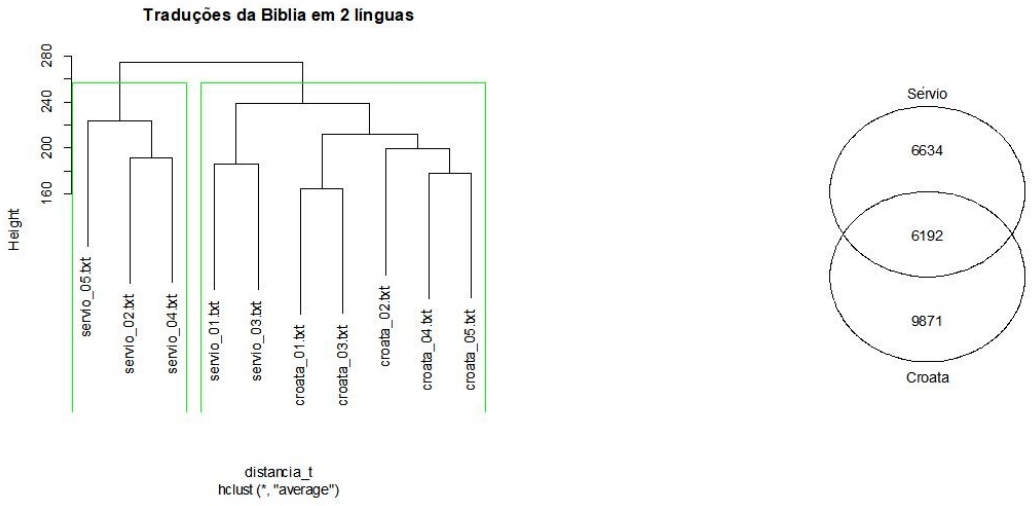
Fonte: o autor.

Os índices de *Rand*, *Jaccard* e *Fowlkes&Mallows* indicam que os agrupamentos formados na Figura 5.11 não são iguais aos agrupamentos esperados (i.e., 30 agrupamentos com 5 objetos cada, pois são 30 traduções divididas em 5 arquivos). O índice de *Dunn* ficou abaixo do parâmetro 1.081 padronizado no Quadro 5.3. Se os agrupamentos estivessem conforme o esperado, o índice *Dunn* seria igual ou maior que 1.081. O índice de *Davies-Bouldin* ficou acima de 0.900 (padronizado no Quadro 5.3); para que os agrupamentos formados fossem iguais ou melhores que o esperado, esse valor deveria ser inferior a 0.900. O índice de *Silhouette* ficou abaixo de 1, isso sugere que as formações de agrupamentos não estão conforme o esperado.

Além da percepção visual no dendrograma de que os agrupamentos não foram construídos conforme o esperado (30 grupos com 5 objetos cada), os índices baseados em critérios interno e externo também sugerem que há algo incomum na formação dos agrupamentos. Os índices não apontam onde, mas, como relatado anteriormente, é sabido que são os agrupamentos formados pelos textos em dinamarquês/norueguês e sérvio/croata.

Ao comparar as traduções da Bíblia para as línguas croata e sérvia, os seguintes resultados foram observados:

Figura 5.14: Modelos 01 e 02 aplicados às línguas croata e sérvia.




Fonte: O autor.

A formação de agrupamentos no dendrograma da Figura 5.14 não distinguiu os textos da língua sérvia dos textos da língua croata. Os índices baseados em critérios interno e externo sugerem que os agrupamentos não estão conforme o esperado (i.e., o valor 1 para os índices de *Dunn, Jaccard* e *Fowlkes&Mallows* sugere que os agrupamentos formados e conhecidos são iguais e no Quadro 5.6 os valores não são iguais a um; os indices *Dunn, Davies-Bouldin* e *Silhouette* também não apresentam valores favoráveis para a identificação de agrupamentos conforme era esperado, dois grupos com textos de mesma língua).

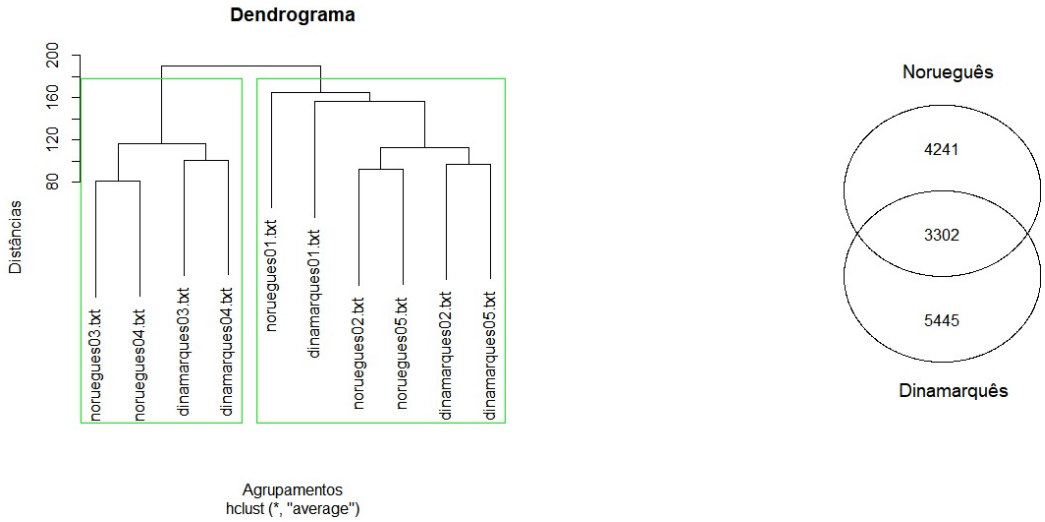
Quadro 5.6: Resultados dos índices do sérvio e do croata.

| **Índices baseados em critérios externos** | | **Índices baseados em critérios internos** | |
|---|---|---|---|
| *Rand* | 0.498 | *Dunn* | 0.744 |
| *Jaccard* | 0.324 | *Davies-Bouldin* | 2.233 |
| *Fowlkes & Mallows* | 0.444 | *Silhouette* | 0.235 |

Fonte: O autor

O modelo de *bi-grama* não atribui aos objetos (i.e., textos) características suficientes para que a análise de agrupamentos diferencie em agrupamentos os textos da língua sérvia e os textos da língua croata. No diagrama de *Venn* da Figura 5.14 observa-se que quase metade das palavras usadas na Bíblia da língua sérvia também aparecem na Bíblia da língua croata. Em relação à Bíblia norueguesa e dinamarquesa foram observados resultados apresentados na Figura 5.15.

Figura 5.15: Modelos 01 e 02 aplicados àss línguas norueguesa e dinamarquesa.




Fonte: O autor.

Quadro 5.7: Resultados dos índices do norueguês e do dinamarquês.

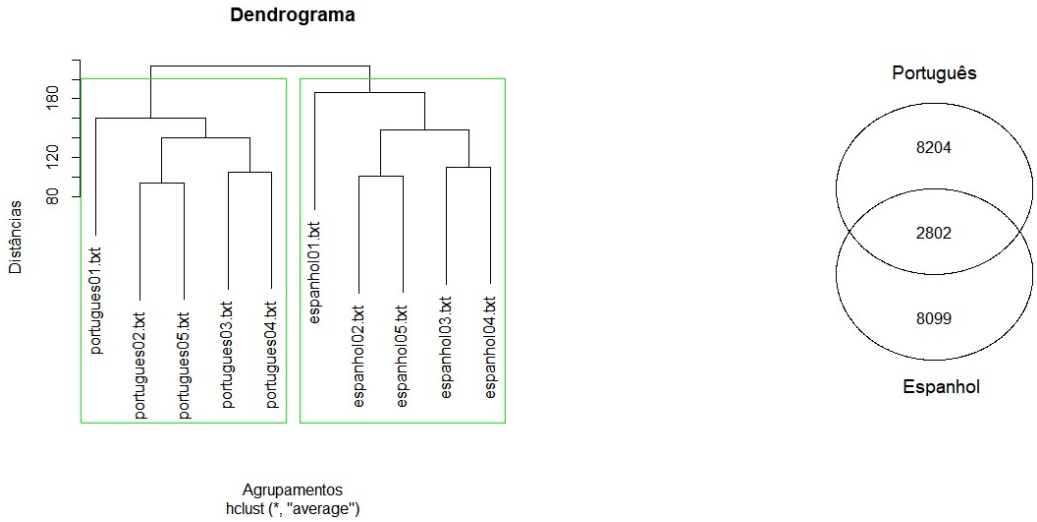
| **Índices baseados em critérios externos** | | **Índices baseados em critérios internos** | |
|---|---|---|---|
| *Rand* | 0.390 | *Dunn* | 0.787 |
| *Jaccard* | 0.242 | *Davies-Bouldin* | 2.131 |
| *Fowlkes & Mallows* | 0.444 | *Silhouette* | 0.330 |

Fonte: O autor.

Os resultados dos textos em dinamarquês e norueguês foram similares aos resultados encontrados entre as línguas sérvia e croata. A proximidade entre as línguas (croata e sérvia de um lado e dinamarquesa e norueguesa do outro) acontece na formação de *bi-gramas* e na quantidade de palavras que elas compartilham. Línguas muito próximas, seja porque elas têm um ancestral comum, ou seja porque elas são variedades de uma mesma língua, apresentam um conjunto de palavras comuns. Com o passar do tempo, a falta de contato entre essas línguas e/ou o contato linguístico delas com outras línguas, acarretam mudanças e, com isso, as línguas que antes eram muito parecidas, passam a apresentar características mais específicas. Por exemplo, português e espanhol têm características semelhantes (como palavras em comum, sintaxe semelhante, ancestrais comuns), mas, no que se refere à formação de *bi-gramas* e conjuntos de palavras semelhantes, apresentam resultados distintos; por sua vez, no que diz respeito aos pares dinamarquês-norueguês e sérvio-croata apresentam *bi-gramas* semelhantes.

A Figura 5.16 e o Quadro 5.8 exibem as análises feitas para a Bíblia em português e espanhol.

Figura 5.16: Modelos 01 e 02 aplicados ao português e ao espanhol.




Fonte: O autor.

Quadro 5.8: Resultados dos índices do português e do espanhol.

| **Índices baseados em critérios externos** | | **Índices baseados em critérios internos** | |
|---|---|---|---|
| *Rand* | 1.000 | *Dunn* | 0.878 |
| *Jaccard* | 1.000 | *Davies-Bouldin* | 2.103 |
| *Fowlkes & Mallows* | 1.000 | *Silhouette* | 0.309 |

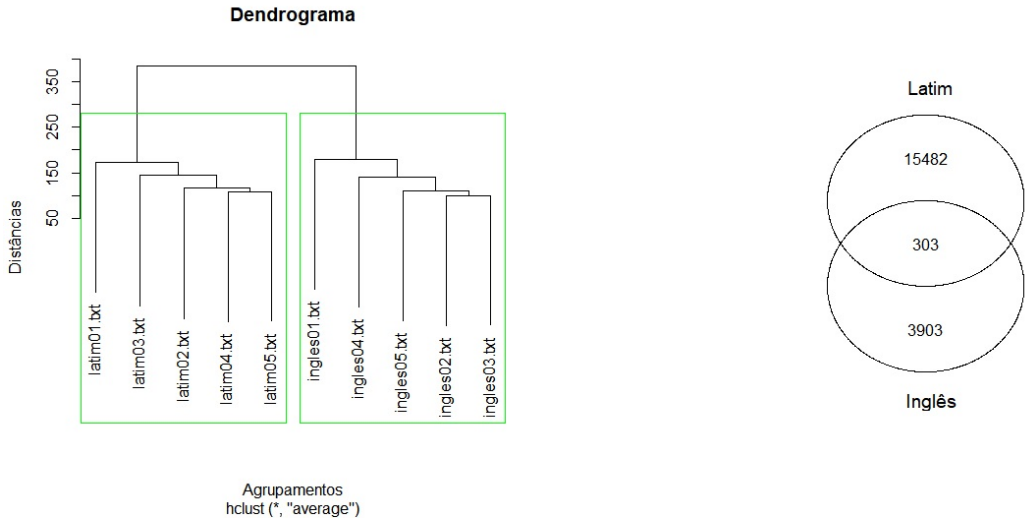
Fonte: O autor.

A proporção de palavras compartilhadas entre o português e o espanhol é menor que entre os pares dinamarquês-norueguês e sérvio-croata. Os índices baseados em critérios externos indicam que todos os agrupamentos gerados pelo dendrograma, para o português e para o espanhol, estão conforme os agrupamentos esperados (2 agrupamentos com 5 objetos cada). Por outro lado, o mesmo não acontece para o dinamarquês-norueguês e o sérvio-croata. Os índices baseados em critérios internos sofreram poucas alterações, se comparados aos índices do norueguês-dinamarquês e sérvio-croata.

Seguem na Figura 5.17 e no Quadro 5.9 os resultados obtidos da comparação entre o latim e o inglês. No dendrograma da Figura 5.17, os agrupamentos formados contêm textos de mesma língua. A quantidade de palavras em comum entre o inglês e o latim é menor (se comparada as quantidades de palavras comuns entre o português e o espanhol) e a distância entre os dois agrupamentos do dendrograma na Figura 5.17 é maior (se comparados ao português-espanhol[8])

[8]Compare as distâncias na escala de valores ao lado esquerda do dendrograma.

Figura 5.17: Modelos 01 e 02 aplicados nas línguas latim e inglês.




Fonte: O autor.

Quadro 5.9: Resultados dos índices do português e do espanhol.

| **Índices baseados em critérios externos** | | **Índices baseados em critérios internos** | |
|---|---|---|---|
| *Rand* | 1.000 | *Dunn* | 1.514 |
| *Jaccard* | 1.000 | *Davies-Bouldin* | 1.186 |
| *Fowlkes & Mallows* | 1.000 | *Silhouette* | 0.620 |

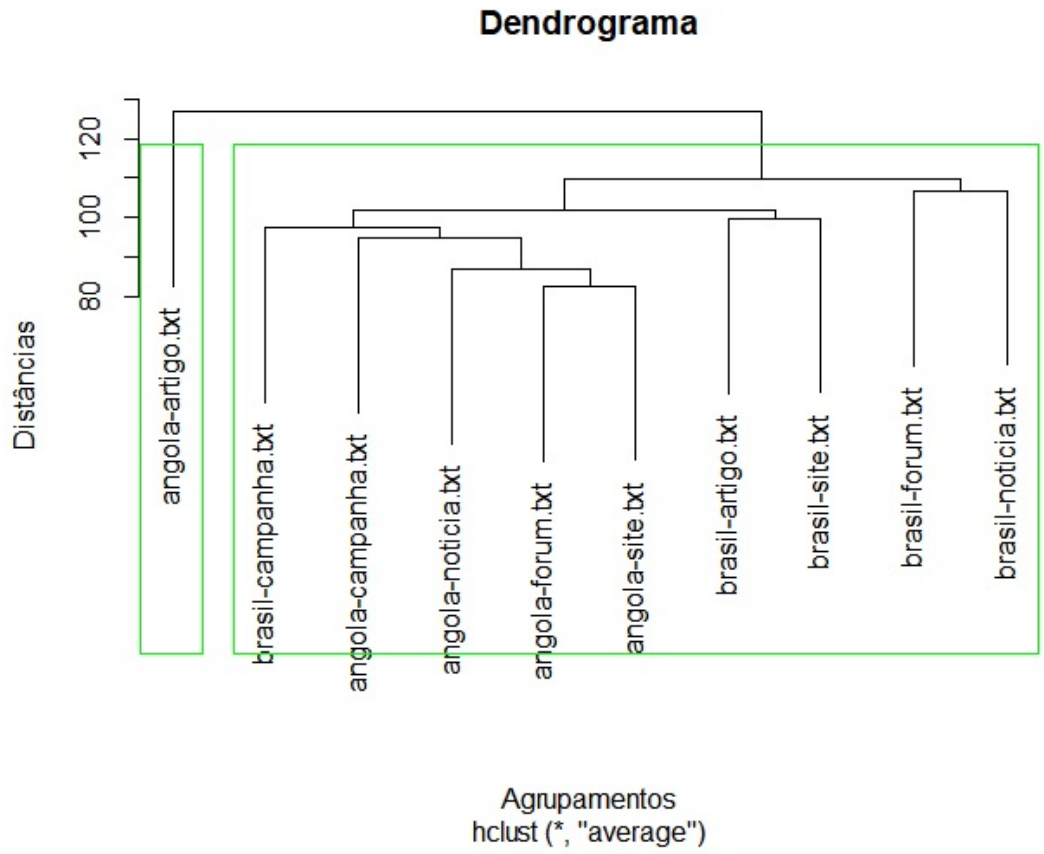
Fonte: O autor.

A semelhança entre a línguas pode ser percebida no dendrograma. Por exemplo, quanto mais próximas duas línguas forem, mais elas tendem a aparecer em um mesmo agrupamento (e.g., as línguas norueguesa-dinamarquesa e servia-croata na Figura 5.13).

O português falado no Brasil e o falado em Angola são variedades de uma mesma língua e, apesar de terem algumas expressões e palavras próprias, elas ainda são línguas muito próximas. Neste momento, é importante chamar a atenção para o fato de que, sendo a Bíblia e a Eneida textos formais (e escritos), é bastante plausível que não se possam encontrar dados das variações do português a partir desse gênero (a escrita tende a ser mais conservadora que a fala; textos escritos na variedade formal das línguas tendem a ser mais conservadores no que se refere à variação). Dessa forma, é importante considerar outros gêneros textuais que permitam, por exemplo, mais influência da fala (onde a variação é mais facilmente percebida). Nesse sentido, para testar a proximidade entre o português brasileiro e angolano, foram considerados textos de sites governamentais, notícias de jornal, artigos, fórum de discussões e campanhas publicitárias (a origem dos textos estão no Apêndice A).

Nos testes com a Bíblia e a Eneida, os arquivos gerados eram de uma única tradução; neste novo teste, com textos em português angolano e brasileiro, pretende-se, além de diversificar os autores e os gêneros textuais, verificar se o Modelo 01 construirá um dendrograma que organize os textos do português brasileiro e angolano separadamente.

Figura 5.18: Dendrograma: português brasileiro e português angolano.




Fonte: O autor.

Quadro 5.10: Resultados dos índices do português brasileiro e do português angolano.

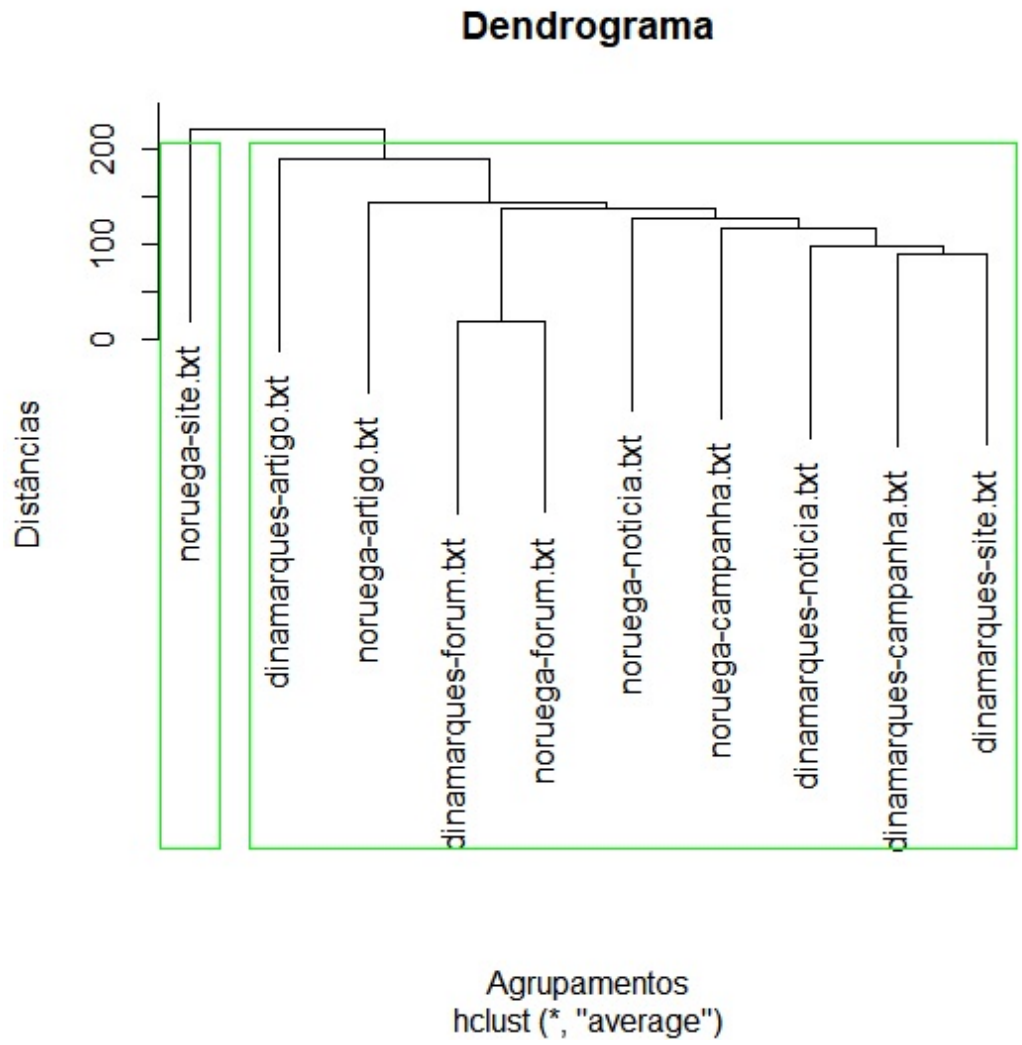
| **Índices baseados em critérios externos** | | **Índices baseados em critérios internos** | |
|---|---|---|---|
| *Rand* | 0.596 | *Dunn* | 0.931 |
| *Jaccard* | 0.400 | *Davies-Bouldin* | 1.073 |
| *Fowlkes & Mallows* | 0.466 | *Silhouette* | 0.172 |

Fonte: O autor.

O dendrograma não reuniu os textos do português angolano em um grupo e do português brasileiro no outro. Os índices baseados em critérios internos estão abaixo de um (se fosse 1 os agrupamentos estariam conforme o esperado).

Com textos em dinamarquês e norueguês retirados da internet de sites governamentais, campanhas publicitárias, artigos, notícias e fóruns de discussão (a origem dos textos estão no Apêndice A), são formados os agrupamentos da Figura 5.19.

Figura 5.19: Dendrograma: dinamarquês e norueguês.




Fonte: O autor.

Quadro 5.11: Resultados dos índices do dinamarquês e do norueguês.

| **Índices baseados em critérios externos** | | **Índices baseados em critérios internos** | |
|---|---|---|---|
| *Rand* | 0.596 | *Dunn* | 0.869 |
| *Jaccard* | 0.400 | *Davies-Bouldin* | 1.150 |
| *Fowlkes & Mallows* | 0.466 | *Silhouette* | 0.324 |

Fonte: O autor.

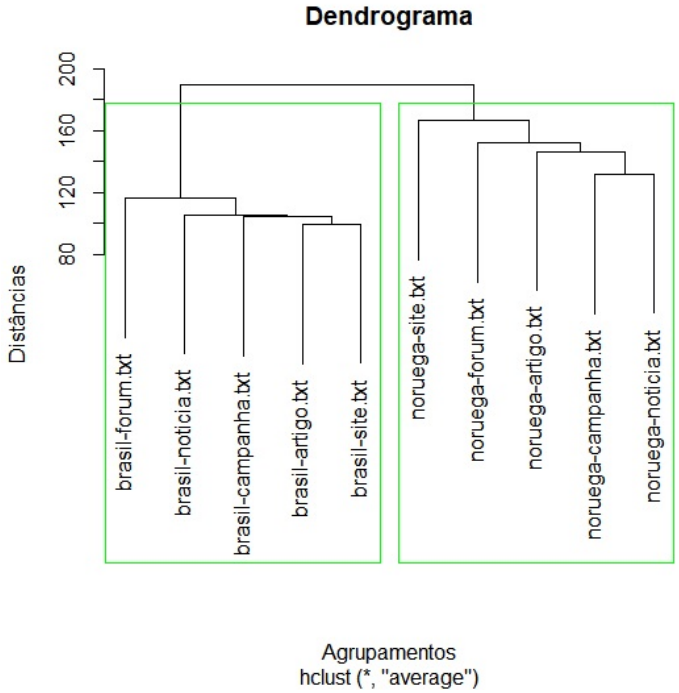
Ao comparar as Figuras 5.18 e 5.19 e os Quadros 5.10 e 5.11, percebe-se que o compor-

tamento entre as variedades do português comportam-se como o dinamarquês. Sandoy (2011), ao descrever brevemente o processo de independência da Noruega em 1814 do reino da Dinamarca, relata que o povo norueguês assumiu como língua oficial o dinamarquês e que, a partir de 1910, iniciaram-se, na Noruega, várias propostas políticas para a criação de uma identidade linguística própria para o país. Com isso, uma série de reformas ortográficas foram implantadas. Historicamente as línguas dinamarquesa e norueguesa são variedades de uma mesma língua.

Com relação à diferença entre as línguas sérvia e croata, Szlarz (2012) relata que, com o final da antiga Iugoslávia, os croatas começaram uma política em prol de uma identidade patriótica e uma língua *materna* própria fortaleceria esse ideal. Por serem falantes de um dialeto da língua sérvia, os croatas iniciaram políticas de reformas ortográficas e a formalização de uma identidade linguística própria[9].

Os agrupamentos de mesma língua acontecem mesmo considerando conjuntos de textos de diferentes gêneros. Na Figura 5.20, o dendrograma agrupa os textos conforme uma mesma língua.

Figura 5.20: Dendrograma: português e norueguês.




Fonte: O autor.

[9]Não está no escopo deste trabalho a discussão mais aprofundada sobre questões sociopolíticas que definem a distinção entre língua e dialeto. Foge também do escopo desse trabalho a discussão sobre o papel de convenções ortográficas na formação de uma língua.

Quadro 5.12: Resultados dos índices do dinamarquês e do norueguês.

| **Índices baseados em critérios externos** | | **Índices baseados em critérios internos** | |
|---|---|---|---|
| *Rand* | 1.000 | *Dunn* | 0.853 |
| *Jaccard* | 1.000 | *Davies-Bouldin* | 2.035 |
| *Fowlkes & Mallows* | 1.000 | *Silhouette* | 0.300 |

Fonte: O autor.

Os resultados gerados pela aplicação do Modelo 1 e do Modelo 2 evidenciam uma característica das línguas naturais: a possibilidade de variação. A variação dentro de uma mesma língua e a proximidade entre línguas (que no passado eram tomadas politicamente como a mesma) são explicitadas no dendrograma (pelos agrupamentos que reúnem textos de diferentes línguas).

## *5.3 Discussões*

As línguas naturais obedecem a certos padrões. As matrizes geradas pelos *heatmaps* (Figura 5.7), por exemplo, exibem padrões de cores semelhantes entre os textos que compartilham uma mesma língua e esse padrão não se repete entre os *heatmaps* de textos de diferentes línguas. Para transformar esses padrões vistos nos *heatmaps* em informação, optou-se por transformar as matrizes em vetores e agrupá-los em uma tabela (e.g., Tabela 5.1); essa tabela serviu de subsídio para o Modelo 01 criar dendrogramas, gráficos em escalonamento multidimensional, gerar índices para validar os agrupamentos formados e verificar se esses padrões podem ser medidos, agrupados e analisados.

Nos dendrogramas das Figuras 5.6 e 5.10, verificasse que todos os agrupamentos formados contêm textos que compartilham uma mesma língua e, sob outra perspetiva, nos gráficos de escalonamento multidimensional das Figuras 5.3 e 5.8, os textos com atributos (i.e., *bi-gramas*) semelhantes aparecem mais próximos entre si, enquanto os menos semelhantes aparecem mais distantes.

Por outro lado, nem todos os textos analisados apareceram no dendrograma em agrupamentos de textos com mesma língua. Nas Figuras 5.13, 5.14 e 5.15 os textos das línguas norueguesa e dinamarquesa apareceram em um mesmo agrupamento e os textos das línguas sérvia e croata apareceram juntos em outro agrupamento. Era esperado que a análise de agrupamentos, por meio de *bi-gramas*, pudesse reunir, em grupos, os textos que compartilham uma mesma língua e foi o que aconteceu.

A variação dentro de uma mesma língua não é distinguída em diferentes agrupamentos

no dendrograma. Na Figura 5.18, a variação linguística nos textos em português produzidos em Angola e no Brasil não é separada em agrupamentos específicos e o mesmo fato aconteceu com os agrupamentos apresentados para as línguas norueguesa e dinamarquesa na Figura 5.19.

Historicamente, a língua norueguesa era uma variação linguística da língua dinamarquesa; e a língua croata era uma variação da língua sérvia. As variações linguísticas de mesma língua se reúnem no dendrograma em um mesmo agrupamento; isso explica o resultado para os pares dinamarquês-norueguês e sérvio-croata.

Os índices baseados em critérios externos sinalizaram corretamente com 1 que os agrupamentos formados correspondiam a grupos com textos de uma mesma língua (e.g., Quadros 5.8, 5.9 e 5.12). Quando, no dendrograma, a maioria dos agrupamentos era formada por conjuntos de textos de mesma língua (i.e., conjuntos com cinco objetos em cada agrupamento) (e.g., Quadro 5.5 e Figura 5.11) os valores dos índices baseados em critérios externos ficaram próximos de 1 (entre 0,83 e 0,99). Quando o dendrograma continha somente agrupamentos com textos de variações linguísticas (e.g., Quadros 5.6, 5.7, 5.10 e 5.11) os índices ficaram abaixo de 0,6.

Dentre os três índices baseados em critérios externos (*Dunn*, *Jaccard* e *Fowlkes&Mallows*), o índice de *Jaccard* sugeriu valores abaixo de 0,5, nos casos em que o dendrograma não distinguia as variedades de uma língua; e valores acima de 0,5, para os casos em que a maioria dos agrupamentos continha textos de línguas distintas. Dessa forma, o índice de *Jaccard* é o que oferece resultados em que a formação de agrupamentos de textos (com línguas distintas e com variedades de uma língua) apresenta-se mais bem delineada, facilitando a distinção entre os dois cenários.

Em relação aos índices baseados em critérios internos, o índice de *Dunn* ofereceu seu melhor resultado no Quadro 5.9, em um cenário onde as semelhanças entre os textos em latim e inglês eram poucas; porém, quando as semelhanças entre as línguas aumentaram, seja porque eram variedades de uma língua ou porque tinham *bi-gramas* semelhantes (e.g. Quadros 5.16 e 5.11), o índice de *Dunn* reduziu o valor dos resultados e, com isso, dificultou a identificação de uma escala de valores que pudessem ser usados para distinguir agrupamentos de textos de mesma língua e textos com variedades de uma língua.

Os índices de *Silhouette* e *Davies-Bouldin* também geraram escalas de valores pouco conclusivas, que não puderam ser usados para distinguir os agrupamentos com textos de variedades de uma língua.

Capítulo Seis

# Considerações finais

## *6.1 Considerações finais*

Linguistas como Bybee (2016) e Greenberg (2005) consideram que a análise de frequências é uma boa técnica para a identificação de padrões recorrentes nas línguas. Apesar de as línguas passarem pelo processo de mudança ao longo do tempo e terem variações, elas apresentam estrutura aparente e regularidade de padrão, ou seja, é possível identificar propriedades comuns às línguas (BYBEE, 2016; GREENBERG, 2005; SAPIR, 2004). Para algumas correntes teóricas, as línguas naturais são cálculos inferenciais uma vez que o tempo todo fazemos uso de raciocínios inferenciais (OLIVEIRA, 2004).

Este trabalho caracteriza-se por ser uma pesquisa em mineração de dados e análise de frequência de pares de letras. De modo mais específico, trata-se da extração de informações de textos de maneira que seja possível agrupá-los a partir de uma língua por eles compartilhada. Esta pesquisa investigou propostas de algoritmos construídos para identificar a língua de um texto. Uma das questões que balizou este trabalho está relacionada à falta de precisão nos algoritmos baseados em *n-gramas* em relação à identificação das línguas, pois, a depender do texto em análise, os algoritmos não apresentam precisão na distinção de textos de algumas línguas, como as línguas norueguesa e dinamarquesa.

A busca pela explicação dessa imprecisão contou com estudos das ciências da linguagem, assumindo-se que as línguas naturais variam (as línguas naturais são conjuntos/feixes de variedades). Esse fato sobre o funcionamento da linguagem ainda não foi devidamente descrito e capturado pelos algoritmos de identificação de línguas.

Em termos gerais, as línguas são faladas por diferentes comunidades de falantes. Essas comunidades podem decidir formalizar politica e socialmente que falam uma língua específica, mesmo que essa nova língua ainda seja uma variedade de outra; é como se os falantes do português brasileiro decidissem que a partir de certo momento seriam falantes do *brasileirês* e não mais do português. Política e socialmente é uma nova língua, mas estruturalmente ela ainda mantêm traços que a caracterizam como uma variedade de outra língua.

A análise de *bi-grama*, apoiada pela análise de agrupamento, neste trabalho, mostrou que é possível agrupar os textos que pertencem a variedades linguísticas muito próximas, mas não consegue distinguir essas variedades em agrupamentos específicos. Se os textos analisados estivessem em alfabeto fonético, ao invés de caracteres do alfabeto latino-

europeu, talvez a análise de agrupamentos revelasse aspectos socioculturais (i.e., fatores extralinguísticos) das variedades presentes nos textos. Estudos em sociolinguística revelam que existe diferença nos modos como os falantes de mesma língua falam, dependendo de sua classe social, regiões, faixa etária, por exemplo. Esta pesquisa não investigou textos transcritos com o alfabeto fonético.

As línguas podem ser classificadas e categorizadas de diferentes maneiras[1]; os algoritmos do Modelos 01 e 02 são aplicados em textos com caracteres do alfabeto latino-europeu, para estudos interessados em verificar a proximidade entre línguas.

## 6.2 Contribuições

Os resultados apresentados neste trabalho indicam que, por meio de *bi-gramas*, a identificação e o agrupamento de textos em conjuntos que compartilham uma mesma língua são possíveis quando as línguas envolvidas não são variedades de uma mesma língua. Por outro lado, as variedades de mesma língua se reúnem em agrupamentos próprios. Os algoritmos dos Modelos 01 e 02 serão úteis em pesquisas relacionadas a estudos interessados em agrupar as variedades de uma mesma língua. Línguas muito próximas podem revelar contato linguístico entre comunidades de falantes; quando essas comunidades se isolam ou perdem o contato, as línguas dessas comunidades tendem a sofrer influência e a se modificam de formas diferentes, podendo manter traços comuns, ou não. Os algoritmos dos Modelos 01 e 02 ajudarão inferindo se essas línguas são próximas a ponto de seres variedades de uma mesma língua.

## 6.3 Atividades futuras de pesquisa

Em trabalhos futuros, pretende-se usar a teoria de redes complexas para desenvolver um algoritmo que crie redes semânticas e calcule as probabilidades de as línguas analisadas serem variedades/dialetos (de uma língua). A pesquisas futuras, também caberá a análise de textos transcritos no alfabeto fonético internacional, considerando aspectos linguísticos e extralinguísticos (e.g., como tempo, espaço, classe social e faixa etária).

[1] Como os exemplos comentados no Capítulo 2.

Apêndice A

# Origem dos textos

Segue os sites da Internet de onde os textos analisados foram retirados:

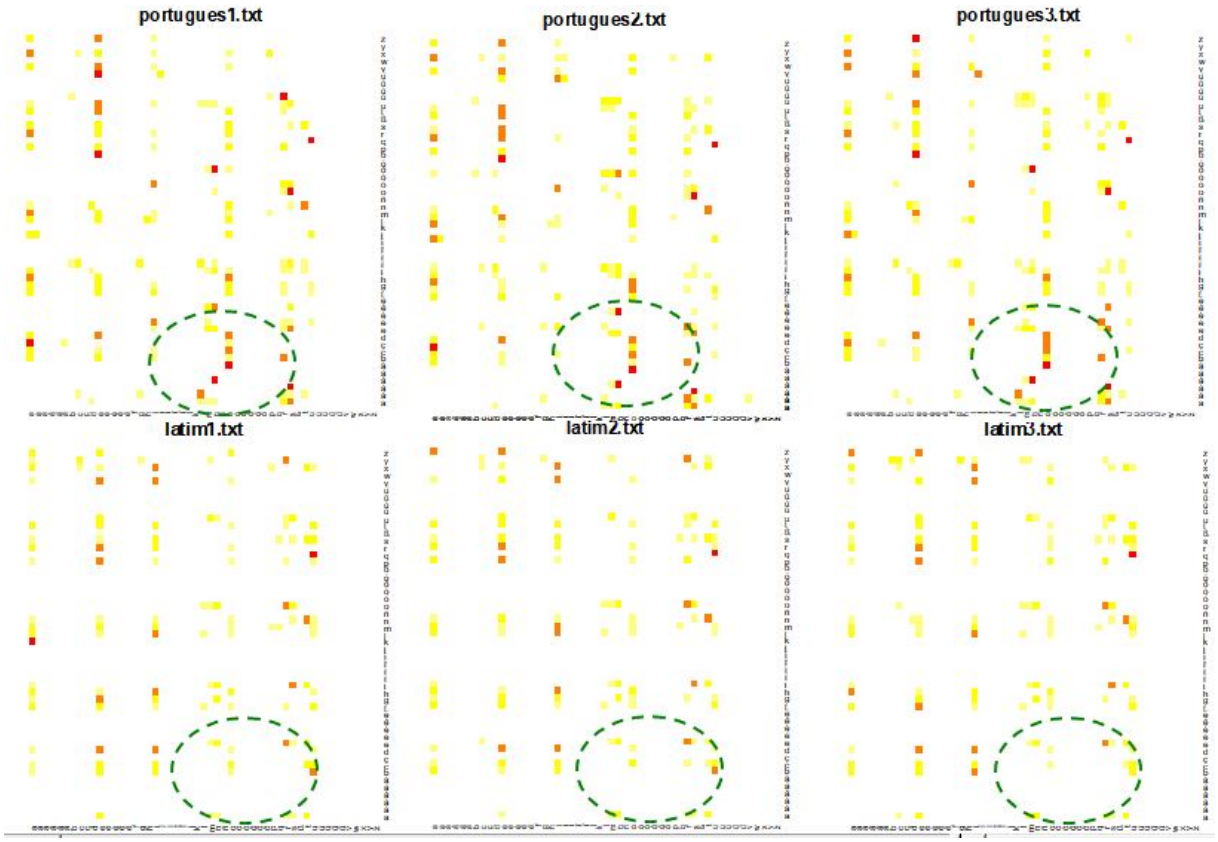
| **Livros / Línguas** | **Sites** |
|---|---|
| Bíblia (Novo Testamento) | https://newchristianbiblestudy.org/<br>http://worldbibles.org/ |
| Eneida | https://www.gutenberg.org<br>http://www.edu.mec.gub.uy/biblioteca_digital/libros/V/Virgilio%20-%20La%20Eneida%20(en%20prosa).pdf<br>https://www.ebooksgratuits.com/html/virgile_eneide.html<br>http://www.gottwein.de/Lat/verg/aen01de.php<br>http://www.ebooksbrasil.org/eLibris/eneida.html |
| Norueguês | Forum: https://www.diskusjon.no/index.php? showforum=276<br>Notícia: www.aftenbladet.no<br>Site: https://www.regjeringen.no<br>Artigo: http://www.ub.umu.se/sites/default/files/dokument /tjanstesidor/Vad_vet_artikel2.pdf<br>https://www.coca-cola.no/presse/press-releases/coca-cola-fortsetter-sin-sukkerfrie-reise-lanserer-kun-nyheter-uten-sukker |
| Dinamarquês | Artigo:http://www.sdu.dk/-/media/files/information_til/studerende_ved_sdu/din_uddannelse/idraet/aktuel_studieordning/kandidatstudieordning_idraet_sundhed_opdt290916.pdf<br>Notícia: http://ekstrabladet.dk/om_ekstra_bladet/den_noedvendige/ historie/article4413269.ece<br>Forum: https://www.hardwareonline.dk/traad.aspx<br>Site: http://stm.dk/_p_14591.html<br>Campanha publicitária: https://www.coca-cola.dk/stories/derfor-er-du-mere-glad-om-sommeren |

| **Livros / Línguas** | **Sites** |
|---|---|
| Português Brasileiro | Artigo: http://www.ufrgs.br/deds/copy_of_imagens/Manual%20Artigo%20Cientifico.pdf<br>Forum: http://Forum.techtudo.com.br/perguntas/78316/android-ou-ios-qual-e-o-melhor?ordenacao=votos&pagina=2<br>Notícia: https://g1.globo.com/politica/Notícia/turma-do-stf-rejeita-recurso-e-mantem-condenacao-de-maluf-por-lavagem-de-dinheiro.ghtml<br>Site: http://www.brasil.gov.br/servicos/perguntas-frequentes<br>Campanha publicitária: https://www.cocacolabrasil.com.br/ historias/ oito-dicas-valiosas-para-uma-entrevista-de-emprego |
| Português Angolano | Notícia: http://www.angoNotícias.com/<br>Forum: http://www.angoNotícias.com/<br>Site: http://www.minct.gov.ao/<br>Artigo: ras.revues.org<br>Campanha publicitária: http://www.welcometoangola.co.ao/_campanha _cocacola _troca _marca _pelo _nome_dos_consumidores |

Apêndice B

# Figuras ampliadas

Figura B.1: Comparação entre as frequências de mesma língua




Fonte: O autor.

Figura B.2: Parte de uma Matriz de Custo

| | a | á | à | â | ä | ã | b | c | ç | d | e | é | è | ê | ë | f | g | h | i | í | ì | î | ï | j | k | l | m | n |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **a** | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 14 | 3 | 12 | 0 | 0 | 0 | 0 | 0 | 1 | 7 | 0 | 17 | 1 | 0 | 0 | 0 | 0 | 0 | 20 | 14 | 25 |
| **á** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 |
| **à** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **â** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |
| **ä** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **ã** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **b** | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 |
| **c** | 15 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 14 | 2 | 0 | 1 | 0 | 0 | 0 | 4 | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 |
| **ç** | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **d** | 20 | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 35 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| **e** | 3 | 0 | 0 | 0 | 0 | 0 | 2 | 4 | 4 | 3 | 0 | 0 | 0 | 0 | 0 | 2 | 6 | 0 | 13 | 0 | 0 | 0 | 0 | 1 | 0 | 15 | 34 | 34 |
| **é** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **è** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **ê** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 4 |
| **ë** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **f** | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 6 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **g** | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| **h** | 9 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **i** | 25 | 0 | 0 | 0 | 0 | 1 | 4 | 6 | 1 | 5 | 1 | 0 | 0 | 0 | 0 | 1 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 11 | 18 |
| **í** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| **ì** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **î** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **ï** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **j** | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **k** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **l** | 15 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 10 | 1 | 0 | 1 | 0 | 0 | 2 | 11 | 15 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| **m** | 20 | 0 | 0 | 0 | 0 | 1 | 3 | 0 | 0 | 0 | 25 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **n** | 11 | 1 | 0 | 0 | 0 | 8 | 0 | 9 | 0 | 22 | 11 | 4 | 0 | 0 | 0 | 3 | 3 | 6 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

Fonte: O autor.

Apêndice C

# Algoritmos em R para calcular distância

### a) Buscar distância entre dois objetos

```
#
#Equação: Euclidiana
#
Eq_Euclidiana <- function(V1,V2){
    Eq_Euclid <- function(v1,v2){
        if( length(v1) == 1){
            return( (v1[1]-v2[1])^2 )
        } else {
            return(Eq_Euclid(v1[-1],v2[-1]) + (v1[1]-v2[1])^2)
        }
    }
    return( sqrt(Eq_Euclid(V1,V2)) )
}
```

```
#
#Equação: Manhattan
#
Eq_Manhattan <- function(v1,v2){
    if( length(v1) == 1){
        return( abs(v1[1]-v2[1]) )
    } else {
        return(Eq_Manhattan(v1[-1],v2[-1]) + abs(v1[1]-v2[1]))
    }
}
```

```
#
#Equação Canberra
#
Eq_Canberra <- function(v1,v2){
    if( length(v1) == 1){
        return( abs(v1[1]-v2[1])/(abs(v1[1])+abs(v2[1])) )
    } else {
        return(Eq_Canberra(v1[-1],v2[-1]) + (abs(v1[1]-v2[1])/(abs(v1[1])+abs(v2[1]))) )
    }
}
```

## b) Cálculo de Distância entre objetos

```
#
#Métodos para Cálculo de Distâncias
#
Calculo_De_Distancias <- function(tabela, Metodo) {
     tot_lin <- nrow(tabela)
    tot_comb <- factorial(tot_lin) / (factorial(tot_lin-2)*factorial(2))
    tab_dist <- matrix(nrow=tot_comb,ncol=3)
   tab_dist[,] <- 0
         pos <- d <- 0
    for( a in 1:(tot_lin-1) ) {
        for( b in (a+1):(tot_lin) ) {
            v1 <- as.vector(tabela[a,2:ncol(tabela)])
            v2 <- as.vector(tabela[b,2:ncol(tabela)])
            if(Metodo == 1) {d <- Eq_Euclidiana(v1,v2) }
            if(Metodo == 2) {d <- Eq_Manhattan(v1,v2)}
            if(Metodo == 3) {d <- Eq_Canberra(v1,v2) }
            pos <- pos + 1
            tab_dist[pos,1] <- a
            tab_dist[pos,2] <- b
            tab_dist[pos,3] <- d
        }
    }

    #Classificar distancia de modo crescente
       ordem <- order(tab_dist[,3], decreasing = FALSE)
    tab_dist <- tab_dist[ordem,]
    colnames(tab_dist) <- c("Objeto01","Objeto02","distancias")
    return(tab_dist)
}

#Distâncias: 1-Euclidiana;2-Manhattan;3-Canberra;
Metodo <- 1
#tabela: Tabela com os objetos e atributos
tab_dist <- Calculo_De_Distancias(tabela,Metodo)
```

```
#
#Retorna a distância entre dois objetos
#
Distancia_Entre_Dois_Objetos <- function(V1,V2, tab_dist){
    for( i in 1:nrow(tab_dist) ){
        if( ( V1 == tab_dist[i,1] & V2 == tab_dist[i,2]) | (V1 == tab_dist[i,2] & V2 == tab_dist[i,1]) ) {
            return ( tab_dist[i,3] )
        }
    }
}
```

```
#
#Retona um vetor com os grupos identificados
#
ListarGrupos <- function(v){
    v_resp <- v[1]
    for( i in 1:length(v) ){
        ctrl <- 0
        for ( j in 1:length(v_resp) ){
            if ( v[i] == v_resp[j] ) {
                ctrl <- 1
                break
            }
        }
        if (ctrl ==0) {
            v_resp <- cbind(v_resp, v[i])
        }
    }
    return(as.vector(v_resp))
}
```

```
#
# Retorna os objetos, as formações de grupos e respectivas distâncias entre grupos
#
GerarGrupos <- function(tabela, metodo_intragrupo, metodo_intergrupos){
    tab_dist <- Calculo_De_Distancias(tabela, metodo_intragrupo )
     tot_obj <- nrow(tabela);    tot_lin <- nrow(tab_dist) ;  G <- c(-1:-(tot_obj+1))
    G[length(G)] <- 0;  tab_grup <- rbind(G); a <- b <- ct <- 0
    while(b < tot_lin){
        extr <- 0
        a <- nrow(tab_grup); b <- b + 1; G <- tab_grup[a,]; obj1 <- tab_dist[b,1]; obj2 <- tab_dist[b,2] ; dist <- tab_dist[b,3]
        if(tab_grup[a,obj1] < 0 & tab_grup[a,obj2] < 0) {
              ct <- ct + 1; G[obj1] <- G[obj2] <- ct; G[tot_obj+1] <- dist
        } else {
                g <- ListarGrupos(G[1:(length(G)-1)])
            if(length(g) == 1) {break}
                g_pares <- ListarParesDeGrupos(g)
            md <- ProximoGrupo(tab_dist, g_pares, G[1:(length(G)-1)], metodo_intergrupos)
            if(md[1]<0 & md[2]<0){extr <- 1}
            if(md[1]<0 | md[2]<0){
                if(md[1]<0) {
                    z <- md[2]; w <- md[1]
                } else {
                    z <- md[1]; w <- md[2]
                }
                for( i in 1:(length(G)-1) ){
                    if(G[i] == w) {G[i] <- z}
                }
                G[length(G)] <- md[3]
            } else {
                if(md[1]<md[2]) {
                    z <- md[1]; w <- md[2]
                } else {
                    z <- md[2]; w <- md[1]
                }
                for( i in 1:(length(G)-1) ){
                    if(G[i] == w) {G[i] <- z}
                }
                G[length(G)] <- md[3]
            }
        }
        if(extr == 0) { tab_grup <- rbind(tab_grup,G) }
    }
    return (tab_grup)
}
metodo_intragrupo  <- 1 #1 - distância mínima; 2 - Máxima; 3 - Média;
metodo_intergrupos <- 2 #1 - distância máxima; 2 - Média;
tab_grup <- GerarGrupos(tabela, metodo_intragrupo,  metodo_intergrupos)
```

## c) Indices baseados em critérios externos: *Rand*, *Jaccard* e *Fowlkes & Mallows*

```
#
# Retorna os índices de Rand, Jaccard ou Falkes-Mallows
#
IndicesExternos <- function(p,g,ind){
    vl <- c(1:length(p))
    ml <- ListarParesDeGrupos(vl)
    m2 <- matrix(nrow = nrow(ml), ncol = 6)
    for(i in 1:nrow(ml)){
        m2[i,1] <- ml[i,1]
        m2[i,2] <- ml[i,2]
        m2[i,3] <- m2[i,4] <- m2[i,5] <- m2[i,6] <- 0
    }
    colnames(m2) <- c("Grp01","Grp02","a","b","c","d")
    for(i in 1:nrow(m2)){
        #a-Pares que aparecem em g e em p
        if( (p[m2[i,1]] == p[m2[i,2]]) &
                  (g[m2[i,1]] == g[m2[i,2]]) &
                  (g[m2[i,1]] == p[m2[i,1]]) &
                  (g[m2[i,2]] == p[m2[i,2]])   ) { m2[i,3] <- 1 }
        #
        #b-Pares que aparecem em g e não aparecem em v
        if( (g[m2[i,1]] == g[m2[i,2]]) &
            (p[m2[i,1]] != p[m2[i,2]])   ) { m2[i,4] <- 1}
        #
        #c-Pares que aparecem em p e não aparecem em g
        if( (g[m2[i,1]] != g[m2[i,2]]) &
            (p[m2[i,1]] == p[m2[i,2]])   ) { m2[i,5] <- 1}
        #
        #d-Pares que são diferentes entre p e g
        if( (g[m2[i,1]] != g[m2[i,2]]) &
            (p[m2[i,1]] != p[m2[i,2]])   ) { m2[i,6] <- 1}
    }

    a <- b <- c <- d <- 0
    for(i in 1:nrow(m2)){
        a <- a + m2[i,3]
    for(i in 1:nrow(m2)){
        a <- a + m2[i,3]
        b <- b + m2[i,4]
        c <- c + m2[i,5]
        d <- d + m2[i,6]
    }

    if (ind == 1) {indice <-     ( (a+d)/(a+b+c+d) ) }
    if (ind == 2) {indice <-     (    a /(a+b+c)   ) }
    if (ind == 3) {indice <- sqrt(   (a /(a+b) )*( a /(a+c) ) )}

    return(indice)
}

     G <- as.vector( G[5,1:(ncol(G)-1)] )
     P <- c(2,2,2,1,1,1)
indice <- 3 #1-Rand;2-Jaccard;3-FolwkerMallows
IndicesExternos(P,G,indice)
```

## d) Índices baseados em critérios internos

```
#
# Retorna a distância intergrupos usando os métodos min, max ou media
#
DistanciaEntreGrupos <- function(tabela,G,met_intergrupos){

    GravarDist <- function(matriz,g1,g2, valor){
        for(i in 1:nrow(matriz)){
            if( (matriz[i,1]==g1 & matriz[i,2]==g2) |
                    (matriz[i,1]==g2 & matriz[i,2]==g1)   ) {
                matriz[i,3] <- valor
                matriz[i,4] <- matriz[i,4] + 1
                break
            }
        }
        return (matriz)
    }

    distEucl <- Calculo_De_Distancias(tabela, 1)
         vg  <- ListarGrupos(G)
        pg <- ListarParesDeGrupos(vg)
             m <- matrix(nrow=nrow(pg),ncol=4)
    for(i in 1:nrow(pg)){
        m[i,1] <- pg[i,1]
        m[i,2] <- pg[i,2]
        m[i,3] <- 0
        m[i,4] <- 0
    }

    for(i in 1:nrow(distEucl)){
        obj01 <- distEucl[i,1]
        obj02 <- distEucl[i,2]
        if(G[obj01] != G[obj02]){
            distancia <- Distancia_Entre_Dois_Objetos(G[obj01],G[obj02],m)
            if( (met_intergrupos==1) ){
                if( distancia==0 | distEucl[i,3] < distancia){
                    m <- GravarDist(m,G[obj01],G[obj02], distEucl[i,3])
                }
            }
            if( (met_intergrupos==2) ){
                if( distEucl[i,3] > distancia){
                    m <- GravarDist(m,G[obj01],G[obj02], distEucl[i,3])
                }
            }
            if( (met_intergrupos==3) ){
                distancia <- distancia + distEucl[i,3]
                m <- GravarDist(m,G[obj01],G[obj02], distancia)
            }
        }
    }
    if( (met_intergrupos==3) ){
        for(i in 1:nrow(m)){
            m[i,3] <- (m[i,3]/m[i,4])
        }
    }
    colnames(m) <- c("Grupo01","Grupo02","Distancia","QtdeDeDistancias")
    return (m)
}

met_intergrupos <- 3 #min,max,med
G <- c(1,1,2,2,3,3)
DistanciaEntreGrupos(tabela,G,met_intergrupos)
```

```
#
# Retorna o valor da dispersao nos grupos por meio dos métodos media ou max
#
DispersaoIntraGrupos <- function(tabela,G,met_intragrupo){
    distEucl <- Calculo_De_Distancias(tabela, 1)
         vg  <- ListarGrupos(G)
             m <- matrix(nrow=length(vg),ncol=2)
        m[,] <- 0
    for(i in 1:nrow(distEucl)){
        obj01 <- distEucl[i,1]
        obj02 <- distEucl[i,2]
        if(G[obj01] == G[obj02]){
            m[G[obj01],2] <- m[G[obj01],2] + 1
            if( (met_intragrupo==1) & (m[G[obj01],1]<distEucl[i,3]) ){
                m[G[obj01],1] <- distEucl[i,3]
            }
            if(met_intragrupo==2){
                m[G[obj01],1] <- m[G[obj01],1] + distEucl[i,3]
            }
        }
    }
    if(met_intragrupo==2){
        for(i in 1:nrow(m)){
            if(m[i,1]!=0){
                m[i,1] <- (m[i,1] / m[i,2])
            }
        }
    }
    colnames(m) <- c("Dispersão","Quantidade")
    return (m)
}
met_intragrupo<- 1 #max,med
G <- c(1,1,2,3,3,3)
DispersaoIntraGrupos(tabela,G,met_intragrupo)
```

```
#
# Retorna o índice de Dunn
#

IndiceDeDunn <- function(tabela,G,met_intragrupo,met_intergrupos){
   distGG <- dispGk <- indice <- 0
    vg  <- ListarGrupos(G)
     pg <- ListarParesDeGrupos(vg)
       ml <- matrix(nrow=nrow(pg),ncol=4)
       m2 <- matrix(nrow=length(vg),ncol=2)

    ml <- DistanciaEntreGrupos(tabela,G,met_intergrupos)
    m2 <- DispersaoIntraGrupos(tabela,G,met_intragrupo)

    for(i in 1:nrow(ml)){
        if( ml[i,3] > 0){
            if( (distGG==0) | (ml[i,3]<distGG) ){
                distGG <- ml[i,3]
            }
        }
    }
    for(i in 1:nrow(m2)){
        if(m2[i,1]>0){
            if( (dispGk==0) | (m2[i,1]>dispGk) ){
                dispGk <- m2[i,1]
            }
        }
    }
    return (as.double(distGG/dispGk))
}

met_intragrupo  <- 1 #max, med
met_intergrupos <- 1 #min,max,med
G <- c(1,2,2,3,3,3)
IndiceDeDunn(tabela, G, met_intragrupo, met_intergrupos)
```

```
#
# Retorna o índice de Davies-Bouldin
#
IndiceDeDaviesBouldin <- function(tabela,G,met_intragrupo,met_intergrupos){
   distGG <- dispGk <- indice <- 0
    vg  <- ListarGrupos(G)
     pg <- ListarParesDeGrupos(vg)
       ml <- matrix(nrow=nrow(pg),ncol=4)
       m2 <- matrix(nrow=length(vg),ncol=2)
      k <- length(vg)

    ml <- DistanciaEntreGrupos(tabela,G,met_intergrupos)
    m2 <- DispersaoIntraGrupos(tabela,G,met_intragrupo)

    m3 <- matrix(nrow=nrow(pg),ncol=6)
    m3[,] <- 0
    colnames(m3) <- c("Grupo01","Grupo02","Disp01","Disp02","DistG1G2","RelG1G2")

    for(i in 1:nrow(ml)){
        m3[i,1] <- ml[i,1]
        m3[i,2] <- ml[i,2]
        m3[i,3] <- m2[(ml[i,1]),1]
        m3[i,4] <- m2[(ml[i,2]),1]
        m3[i,5] <- ml[i,3]
        if( (m3[i,3]+m3[i,4]==0) | (m3[i,5]==0) ) {
            m3[i,6] <- 0
        } else {
             m3[i,6] <- ((m3[i,3]+m3[i,4]) / m3[i,5])
        }
    }

    RelGG <- valor <- 0
    for(i in 1:k){
        valor <- 0
        for(j in 1:nrow(m3)){
            if( m3[j,1] == i | m3[j,2]==i){
                if( m3[j,6] > valor ) {
                    valor <- m3[j,6]
                }
            }
        }
        RelGG <- RelGG + valor
    }

    return( as.double((1/k)*RelGG) )
}

met_intragrupo  <- 1 #max, med
met_intergrupos <- 2 #min,max,med
G <- c(1,2,2,3,3,3)
IndiceDeDaviesBouldin(tabela, G, met_intragrupo, met_intergrupos)
```

```
#
# Retorna o índice de Silhouette
#
IndiceDeSilhouette <- function(tabela,G){
            n <- nrow(tabela)
         vg  <- ListarGrupos(G)
          ml <- matrix(nrow=length(vg),ncol=4)
       ml[,] <- 0
  colnames(ml) <- c("Total","Qtde","Media","Participantes")

         m2 <- matrix(nrow=n,ncol=9)
        m2[,] <- 0
    grupoprox <- menordist <- paresEmG <- tmp <- 0
    for(i in 1:n){
        menordist <- paresEmG <- grupoprox <- 0
        ml[,] <- 0
        for(j in 1:length(G)){
            if(G[i]!=G[j]){
                 V1 <- as.vector(tabela[i,2:ncol(tabela)])
                 V2 <- as.vector(tabela[j,2:ncol(tabela)])
                   tmp <- Eq_Euclidiana(V1,V2)
                ml[G[j],1] <- ml[G[j],1] + tmp
                ml[G[j],2] <- ml[G[j],2] + 1
                if(ml[G[j],1]!=0 & ml[G[j],2]!=0) {
                    ml[G[j],3] <- ml[G[j],1] / ml[G[j],2]
                }
            }
            ml[G[j],4] <- ml[G[j],4] + 1
        }
        for(j in 1:nrow(ml)){
            if(ml[j,3] != 0){
                if( (menordist==0) | (ml[j,3]<menordist) ){
                    menordist <- ml[j,3]
                    grupoprox <- j
                }
            }
        }
        m2[i,1] <- grupoprox
        m2[i,2] <- menordist
        m2[i,8] <- ml[G[i],4]
        m2[i,9] <- G[i]
    }
    colnames(m2) <- c("GrupoProximo","Distancia","ParesNoGrupo","ParesEntreGrupos","a","b",
    "Silhouette","QtdeNoGrupo","Grupo")
    for(i in 1:n){
        tmp <- 0
        for(j in 1:n){
            if( (m2[i,1]==G[j]) & (i != j) ){
                tmp <-  tmp + 1
            }
            if( (G[j]==G[i]) & (i != j) ){
                m2[i,3] <- m2[i,3] + 1
            }
        }
        m2[i,4] <- tmp
    }

    a <- b <- tot_obj <- 0
    for(i in 1:length(G)){
        a <- b <- 0
        for(j in 1:length(G)){
            if(i!=j){
                V1 <- as.vector(tabela[i,2:ncol(tabela)])
                V2 <- as.vector(tabela[j,2:ncol(tabela)])
                if(G[i]==G[j]){
                    a <- a + Eq_Euclidiana(V1,V2)
                }
                if(G[i]!=G[j] & G[j]==m2[i,1]){
                    b <- b + Eq_Euclidiana(V1,V2)
                }
            }
        }
        if( (m2[i,3] != 0) & (a!=0) ){
            m2[i,5] <- a/m2[i,3]
        }
        if( (m2[i,4] != 0) & (b!=0) ){
            m2[i,6] <- b/m2[i,4]
        }

        if(m2[i,5]>m2[i,6]){
            m2[i,7] <- (m2[i,6] - m2[i,5]) / m2[i,5]
        } else {
            m2[i,7] <- (m2[i,6] - m2[i,5]) / m2[i,6]
        }
        if(m2[i,8]==1){
            m2[i,7] <- 0
        }
    }

    Isil <- 0
    for(i in 1:nrow(m2)){
        Isil <- Isil + m2[i,7]
    }
    ordem <- order(m2[,9], decreasing = FALSE)
       m2 <- m2[ordem,]
    return(Isil/n)
}
G <- c(1,1,1,1,2,2)
```

## e) Modelo 01

```
#
# Retorna o dendrograma com os agrupamentos de texto de mesma língua
#
library(tm)
Modelo01 <- function(endereco,dist_obj,dist_grp,es_multi,heapmaps) {
    docs <- Corpus(DirSource(endereco,encoding="UTF-8"))
    listaTXT <-list();    letras <- vector();     letras <- c("a","b");
         tmp <- 0;         a <- " "
    for( i in 1:length(docs) ){
                txt <- removePunctuation(docs[[i]]$content[1])
                txt <- removeNumbers(txt)
                txt <- stripWhitespace(txt)
                txt <- tolower(txt)
        listaTXT[i] <- txt
        for( j in 1:nchar(txt) ){
            a <- substr(txt, j, j)
            if(a==" " | is.null(a)){} else {
                if(is.element(a,letras)) { } else {
                    letras <- c(letras,a)
                }
            }
        }
    }
    id <- order(letras, decreasing = FALSE)
        letras <- letras[id];   letras <- letras[2:length(letras)]
    nome_arquivo <- vector();   listaMAT <-list()
    matriz_agrup <- matrix();   m_inicial <- table(letras,letras)
         m_inicial[,] <- 0;   tmp <- i <- 1
    for (i in 1:length(docs)) {
                m_inicial[,] <- 0;   nome_arquivo[i] <- as.character(docs[[i]]$meta$id)
        n <- nchar(listaTXT[[i]]);   txt <- listaTXT[[i]]
        for(j in 1:(n-1)){
            for(k in (j+1):(j+1)){
                a <- substr(txt, j, j); b <- substr(txt, k, k)
                if(is.element(a,letras) & is.element(b,letras)) {
                    m_inicial[a,b] <- m_inicial[a,b] + 1
                }
            }
        }
        listaMAT[[i]] <- m_inicial
        if(tmp == 1){
            tmp <- 0
            matriz_agrup <- rbind( as.vector(m_inicial) )
        } else {
            matriz_agrup <- rbind( matriz_agrup, as.vector(m_inicial) )
        }
    }

    tabela <- as.matrix(matriz_agrup)
    if(dist_obj == 1){  mDist <- "euclidean"    }
    if(dist_obj == 2){  mDist <- "manhattan"    }
    if(dist_obj == 3){  mDist <- "canberra"     }
    distancias <- dist(tabela, method = mDist)

    if(dist_grp == 1){  mGrupos <- "single"     }
    if(dist_grp == 2){  mGrupos <- "complete"   }
    if(dist_grp == 3){  mGrupos <- "ave"        }
    grupos <- hclust(distancias,method = mGrupos)
    tmp <- 1
    if(length(docs)>50) { tmp <- 0.5 }
    dev.new()
    plot(grupos, ylab="Distâncias", xlab="Agrupamentos", cex = tmp, hang = 1,nome_arquivo, main = "Dendrograma")
    #rect.hclust(grupos, k=2, border="green")
    if(heapmaps == 1) {
        # Retorna o heatmap dos agrupamentos por língua
        require(graphics); require(grDevices)

        for (i in 1:length(docs)) {
            dev.new()
                    ml <- as.matrix(listaMAT[[i]])
            colfunc      <- colorRampPalette(c("white","yellow","red"))
            pretoEbranco <- colorRampPalette(c("gray","Black"))
            heatmap(ml, Rowv = NA, Colv = NA, main = nome_arquivo[i], col=(colfunc(5)), symm = FALSE)
        }
    }
    if(es_multi == 1) {
        # Retorna o Escalomaneto Multidimensional dos agrupamentos de mesma língua
        library(vegan)
        linguas <- vector(length=length(docs))
        for (i in 1:length(docs)) {
            linguas[i] <- substr(nome_arquivo[i], 1, (nchar(nome_arquivo[i])-5) )
        }
        m2 <- monoMDS(distancias, model = "loc"); x <- m2$points[, 1];  y <- m2$points[, 2]
        dev.new()
        plot(m2, choices = c(1,2),  type = "p", xlim = range(x) + c(0, 0.3), ylim = range(y) + c(0, 0.3))
        orditorp(m2,display="specie",col="red",air=1)
        ordihull(m2, groups=linguas,draw="polygon",col="grey90", label=TRUE) #poligono
    }
    library("clv")
    #Indices baseados em critérios externos
    nro <- 1;   cont <- 0;  totObj <- nrow(tabela);  P <- vector(length=totObj )
    for(i in 1:totObj ){
        cont <- cont + 1;   P[i] <- nro
        if (cont == 5) {
            cont <- 0;   nro <- nro + 1
        }
    }
    G <- as.integer(cutree(grupos, k=(totObj/5)))
    P <- as.integer(P)
    conjuntos <- std.ext(P,G)
    FM    <- clv.Folkes.Mallows(conjuntos)
    Jacc  <- clv.Jaccard(conjuntos)
    Rand  <- clv.Rand(conjuntos)
    #Indices baseados em critérios externos
    scatt <- cls.scatt.data(as.matrix(tabela),G,dist="euclidean")
      Dunn  <- as.double(clv.Dunn(scatt,"complete","single"))
    DB <- as.double(clv.Davies.Bouldin(scatt, "complete", "single"))
    library("fpc")
    indices <- cluster.stats(dist(tabela), G)
    Sil <- indices$avg.silwidth
    indicesIE <- rbind(FM, Jacc,  Rand, Dunn, DB, Sil)
    colnames(indicesIE) <- c("Indices")
    return(indicesIE)
}

endereco <- "C:\\testes\\biblia\\"
dist_obj <- 1 #1-Euclidiana; 2-Manhattan; 3-Canberra;
dist_grp <- 3 #1-Mínima; 2-Completa; 3-Média;
es_multi <- 2 #1-Sim;2-Não
heapmaps <- 1 #2-Sim;2-Não
date()
Modelo01(endereco,dist_obj,dist_grp,es_multi,heapmaps)
date()

```

## f) Modelo 02

```
library(tm)
library(qdap)
library(stringi)
library(gsubfn)
require(gplots)

Modelo02 <- function(endereco) {
    textos <- Corpus(DirSource("C:\\testes\\venn\\",encoding="UTF8"))
    listaTEXTOS <- list()
    for(i in 1:length(textos)){
        a <- stri_trans_general(textos[[i]], "Latin-ASCII")
        a1 <- Corpus(VectorSource(a))
        a2 <- TermDocumentMatrix(a1)
        a3 <- a2$dimnames$Terms
        a3 <- removeNumbers(a3)
        a3 <- a3[a3!=""]
        listaTEXTOS[[i]] <- a3
    }

    vetor01 <- vector(length=length(textos))
    for(i in 1:length(textos)){
        vetor01[i] <- textos[[i]]$meta$id
    }
    names(listaTEXTOS) <- vetor01
    dev.new()
    v <- venn(listaTEXTOS)
}

endereco <- "C:\\testes\\venn\\"
Modelo02(endereco)
```

# Referências


AHMED, B.; CHA, S.H.; TAPPERT, C. Language identification from text using n-gram based cumulative frequency addition. *Proceedings of Student/Faculty Research Day, CSIS, Pace University*, p. 12–1, 2004.

ALENCAR, L. F. Línguas formais, gramáticas e autômatos no processamento automático das palavras. In: _____. *"Abordagens computacionais da teoria da gramática"*. Campinas: Mercado de Letras, 2011. p. 13–75.

AMARAL, F. *Introdução à Ciencia de Dados: mineração de dados e big data*. Rio de janeiro: Alta Books Editora, 2016.

BYBEE, J. *Língua, uso e cognição*. São Paulo: Cortez, 2016.

CAZAMIAS, J.; DIXIT, J.; MAREK, M. Large-scale language classification - writing a detector for 200 languages on twitter. *Disponível em: <https://nlp.stanford.edu/courses/cs224n/2015/reports/24.pdf> Acessado em 20 ago. 2017*, 2015.

FARACO, C. *Linguística histórica: uma introdução ao estudo da história das línguas*. São Paulo: Parábola Editorial, 2005.

GREENBERG, J. *Genetic linguistics: essays on theory and method*. New York: Oxford University Press, 2005.

HAIR, J. F. *et al. Análise multivariada de dados*. Porto Alegre: Bookman, 2009.

HALKIDI, M.; BATISTAKIS, Y.; VAZIRGIANNIS, M. On clustering validation techniques. journal of intelligent information system. *Journal of intelligent information system*, v. 17, p. 107–145, 2015.

HITCHCOCK, J. Crossword puzzle letter frequencies. *Digital Commons*, Wyoming, 1979.

LYONS, J. *Lingua(gem) e linguística: uma introdução*. Rio de janeiro: LTC, 2009.

MORENO, M. L. Frequency analysis in light of language innovation: Exploring letter frequencies across time, from the days of old english to the days of now. *Spring*, 2005.

OLIVEIRA, R. P. de. Formalismos na lingüística: uma reflexão crítica. In: _____. *Introdução à Lingüística, fundamentos epistemológicos*. São Paulo: Cortez, 2004.

PANDE, H.; DHAMI, H. S. Mathematical modelling of occurrence of letters and word's initials in texts of hindi. *Mathematical Modelling of Occurrence of Letters and Word's Initials in Texts of Hindi*, v. 7, p. 19–38, 2010.

PETTER, M. *Introdução à Linguística Africana*. São Paulo: Editora Contexto, 2015.

PEZATTI, E. G. O funcionalismo em linguística. In: _____. *"Introdução à linguística – fundamentos epistemológicos"*. São Paulo: Cortez, 2011.

PRETI, D. *Sociolingüística: os níveis de fala: um estudo sociolingüístico do diálogo na literatura brasileira*. São Paulo: Edusp, 2000.

R-CORE, Team. *R: A language and environment for statistical computing [Internet]*. Vienna: R Foundation for Statistical Computing, 2017.

RIZA, B. *et al.* Decryption through the likelihood of frequency of letters. In: _____. *Workshop on Semantic Web and New Technologies*. Mexico: SemWeb2010, 2010.

ROUSSEEUW, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, v. 20, p. 53–65, 1987.

RUSSELL, S.; NORVIG, K. *Inteligência Artificial*. Rio de janeiro: Elsevier, 2013.

SANDOY, H. Language culture in norway: A tradition of questioning standard language norms. In: _____. *T. Kristiansen and N. Coupland (eds.) Standard Languages and Language Standards in a Changing Europe*. Oslo: Novus, 2011.

SAPIR, E. *Language: An introduction to the study of speech*. New York: Courier Corporation, 2004.

SAUSSURE, F. de. *Curso geral de linguística*. São Paulo: Cultrix, 1969.

SILVA, A. S. da. O corpus condiv e o estudo da convergência e divergência entre variedades do português. *Perspectivas sobre a Linguateca / Actas do encontro Linguateca - 10 anos*, p. 25–28, 2008.

SILVA, L. A. da; PERES, S. M.; BOSCARIOLI, C. *Introdução à mineração de dados: com aplicações em R*. Rio de Janeiro: Elsevier Brasil, 2016.

SZLARZ, E. Novas línguas: A luta por fronteiras abandona as armas e vai ao dicionário. *Disponível em: <https://super.abril.com.br/comportamento/novas-linguas/>. Acessado em: 01 out. 17.*, 2012.

TAKCI, H.; SOGUKPINAR, I. Centroid-based language identification using letter feature. *CICLing-2004*, p. 640–648, 2004.

TAN, P.N.; STEINBACH, M.; KUMAR, V. *Introdução ao DATAMINING Mineração de Dados*. Rio de Janeiro: Editora Ciência Moderna, 2009.

TRAVAGLIA, L. C. *O aspecto verbal no português: a categoria e sua expressão*. Uberlância: Edufu, 2016.

VITRAL, L. O papel da freqüência na identificação de processos de gramaticalização. *SCRIPTA*, v. 9, p. 149–177, 2006.

*Análise de Agrupamento: O problema da identificação de línguas em textos por meio de bi-gramas*

Cleônidas Tavares de Souza Júnior

Salvador, Fevereiro de 2018.