Universidade de Brasília
Departamento de Estatística

Uma proposta para a determinação do número de clusters

Allan Vieira de Castro Quadros

Projeto apresentado para obtenção do título
de Bacharel em Estatística.

Brasília
2018

Allan Vieira de Castro Quadros

Uma proposta para a determinação do número de clusters

Orientador:
Prof. Dr. **André Luiz Fernandes Cançado**

Projeto apresentado para obtenção do título de Bacharel em Estatística.

Brasília
2018

AGRADECIMENTOS

Agradeço a Deus pela saúde, vontade e determinação que me concedeu durante todos os dias para que concluísse este curso.

A minha esposa Beatriz, que embarcou comigo nesse desafio e que me apoiou durante todas as vezes em que tivemos de abdicar dos momentos juntos. Sem você, simplesmente não teria sido possível.

A minha avó Lourdes, de quem sempre recebi um amor incondicional.

A meus pais, Cristina e José Airton, a quem devo minha formação, educação e caráter.

A minha irmã Karine.

Ao Professor Dr. André Luiz Fernandes Cançado, meu orientador, pelos ensinamentos transmitidos e pelas ótimas conversas sobre Rock, Jazz, Blues e política.

Aos demais professores do Departamento de Estatística da UnB, em especial aos professores Antônio Eduardo Gomes, Ana Maria Nogales, Bernardo Andrade, Cibele Queiroz, Cira Otiniano, Claudete Ruas, Démerson Polli, Donald Pianto, Eduardo Monteiro, Eduardo Nakano, George von Borries, Gladston Silva, Joanlise Andrade, Juliana Gomes, Lúcio Vivaldi, Maria Teresa Costa e Raul Matsushita.

Aos professores do Departamento de Matemática: Ary Medino, Cátia Gonçalves, Flávia Zapata, Ráderson da Silva e Simone Bruschi.

A professora Carla Castanho do Departamento de Ciência da Computação e ao professor Márcio Vitorino da Faculdade de Ciência da Informação.

Aos meus professores da Unicamp pelo grande incentivo: Antônio Márcio Buainain, José Maria da Silveira, Rui Albuquerque e Walter Belik.

Ao Rafael de Souza, ao Pedro Beaklini e à Maria Luiza Dantas, pela oportunidade ímpar de aprendizado e trabalho que me proporcionaram.

A todos os colegas do Fundo Nacional de Desenvolvimento da Educação (FNDE), em especial àqueles da DIGEF e DIFIN pela ajuda e compreensão nesses anos que tive de conciliar trabalho e estudo.

Aos meus amigos de Mandraka, Amauri, Diego, Henrique, João, Rodrigo e Thomaz.

Aos amigos do Departamento de Estatística com os quais tive a honra e a felicidade de compartilhar os últimos 4 anos, Adolfo Dias, Caio Balena, Eduardo Hellas, Frederico de Lucca, Luiz Paulo Roriz, Pedro Brom e Rômulo Coutinho.

A Salman Khan e à Khan Academy, onde aprendi a gostar de Matemática e a me preparar para esse desafio.

"In God we trust.
All others must bring data."
(William Edwards Deming)

RESUMO

Determinar o número k de grupos nos dados é um problema recorrente na análise de *clusters*. Neste trabalho, propomos um método para a estimação de k utilizando uma função multiobjetivo. Um algoritmo de agrupamento baseado em árvores geradoras mínimas (MST) é executado nos dados reais e em hipercubos simulados a partir de distribuições Uniformes utilizando uma adptação do conceito de *cubic clustering criterion (CCC)*. As estatísticas de teste obtidas para $k = 2, 3, ...n$ *clusters* são então comparadas e determina-se o valor mais adequado de k. Espera-se que soluções espúrias de agrupamento nos dados apresentem valores de estatísticas de teste próximos aos das simulações. Os testes iniciais do algoritmo indicam um bom funcionamento para dados com *clusters* elipsóides e alongados. A presença de outliers e de grupos anelares, por outro lado, dificultam seu funcionamento.

Palavras-chave: análise de clusters; árvores geradoras mínimas; *cubic clustering criterion*; otimização de Pareto.

ABSTRACT

Finding the number k of groups in data is a common problem in cluster analysis. In this study we propose a method to estimate k using a multiobjective function. We run a Minimum Spanning Tree (MST) based clustering algorithm on real data and on hypercubes which are simulated from Uniform distributions using an adaptation of the cubic clustering criterion (CCC) concept. The test statistics obtained for $k = 2, 3, ...n$ clusters are then compared and the most appropriate value of k is determined. Spurious solutions in grouping real data are expected to present test statistics values close to those of simulations. Initial tests indicate a great performance on data with ellipsoid and elongated clusters. On the other hand, the algorithm fails in the presence of outliers and ring shaped groups.

Keywords: *cluster analysis*; *minimum spanning trees*; *cubic clustering criterion*; *Pareto optimization.*

SUMÁRIO

1 INTRODUÇÃO

> An intelligent being cannot treat every object it sees as a unique entity unlike anything else in the universe. It has to put objects in categories so that it may apply its hard-won knowledge about similar objects encountered in the past, to the object at hand.
>
> *Steven Pinker(1997)*
> *apud Everitt et al. (2011)*

O comportamento de agrupar objetos similares é inerente aos seres humanos. Essa característica tem origem tanto na necessidade de facilitar nosso entendimento sobre as coisas quanto no fato de permitir uma comunicação mais clara desse conhecimento.

Um exemplo básico da utilização do conceito de grupos está na linguagem. Utilizamos as palavras para descrever conjuntos de objetos ou situações similares. Ao dizer que determinados animais são cães, gatos, ratos, etc, estamos agrupando estes animais em conjuntos com características específicas que os distinguem dos demais. Por este motivo, Everitt et al. (2011) dizem que dar nomes às coisas e classificá-las em grupos são essencialmente a mesma coisa.

Na Ciência, assim como no cotidiano do ser humano comum, a necessidade de agrupar objetos similares (ou separar aqueles que são diferentes) também se faz presente. Os exemplos vão da Biologia às Finanças, passando pela Astronomia e diversas outras áreas do conhecimento. Busca-se classificar organismos, planetas, galáxias, tipos de cliente e qualquer outro objeto, indivíduo ou fenômeno cuja análise via agrupamento contribua para a resolução de um problema. A essas técnicas científicas que buscam identificar agrupamentos nos dados, denominamos **análise de *clusters***.

Embora o emprego científico da análise de clusters seja relativamente antigo[1], o avanço computacional das últimas décadas, acompanhado da grande atenção que áreas li-

[1]Everitt et al. (2011) fazem referência ao fato de Aristóteles ter elaborado um sistema para classificar e agrupar espécies de animais na Antiguidade, de acordo com o fato de possuírem sangue vermelho ou não (invertebrados). Outros exemplos também são descritos pelos mesmos autores.

gadas à Inteligência Artificial vêm recebendo atualmente, levou a uma ampla disseminação das técnicas de *clustering*. Uma consulta rápida ao *Task View* do **CRAN - The Comprehensive R Archive Network** [2] para o assunto *Cluster* indica a existência de mais de 100 pacotes relacionados a implementação de algoritmos de *clustering* em linguagem **R**, em números de junho de 2018.

Não obstante tal disseminação, um problema comum a quase todas as técnicas disponíveis consiste em **como determinar o número de *clusters*** presentes nos dados. Na maioria dos algoritmos devemos, de antemão, informar um k específico - que diz respeito ao número de grupos presentes nos dados em análise. Contudo, ao utilizarmos técnicas de clusterização diferentes informando um mesmo valor de k, ainda assim provavelmente teremos soluções diferentes para o mesmo conjunto de dados. Segundo Handl e Knowles (2007, p. 56–57) isto se deve ao fato de os algoritmos de *clustering* consistirem na otimização de funções uniobjetivas, o que faz com que o melhor resultado utilizando uma técnica baseada em desvio (soma de quadrados) seja diferente daquele ao se empregar uma técnica baseada em conectividade, por exemplo.

Por esse motivo, este trabalho objetiva propor um método para determinação do número de *clusters* utilizando como base a otimização multiobjetivo do nível de conectividade e da soma de quadrados intra-cluster das soluções geradas. Não é um método tão ambicioso quanto o proposto por Handl e Knowles (2007), mas, ainda sim, é inspirado em sua abordagem. Esses autores propuseram um algoritmo de clusterização que utiliza funções multiobjetivo para encontrar as partições nos dados de forma simultânea à obtenção do número k ideal de *clusters*.

O método aqui proposto consiste, *grosso modo*, na aplicação de uma técnica de clusterização nos dados reais para $k = 2, 3, ..., n$ grupos e obtenção de duas estatísticas de teste para cada uma dessas k soluções. O processo é repetido em dados simulados e, ao final, comparam-se as estatísticas de teste dos dados reais e dos simulados com vistas a definir o número adequado de *clusters*. Como método de clusterização, fora confeccionado um algoritmo auxiliar baseado em *Minimum Spanning Trees (MST)* (ou árvores geradoras mínimas) - descrito na seção 3.1.2. Para as simulações, utilizou-se uma adaptação da ideia de *Cubic Clustering Criterion - CCC* - descrito em 3.2. Como estatísticas de teste, utilizaram-

[2] https://cran.r-project.org/web/views/Cluster.html

se soma de quadrados total dentro dos *clusters* e a conectividade entre os pontos da árvore geradora mínima. Outras técnicas assessórias utilizadas serão descritas também no âmbito da seção 3. Para a implementação do método, utilizou-se a linguagem **R** em conjunto com a linguagem **C++** por meio do pacote **Rcpp** [3].

A organização do texto começa com uma breve revisão bibliográfica sobre as técnicas de análise de *clusters* na seção 2.1 seguida de uma avaliação da importância das técnicas de validação de *clusters* na seção 2.2. O trabalho avança na parte metodológica com a descrição teórica mais detalhada das técnicas empregadas na confecção do método proposto, como MST (seção 3.1), *MST based clustering* (seção 3.1.2), *Cubic Clustering Criterion - CCC* (seção 3.2), e Otimalidade de Pareto (seção 3.3), culminando na apresentação completa do algoritmo desenvolvido para este trabalho na seção 3.4.1. Por fim, são apresentados os resultados obtidos com a aplicação do método (seção 4) aos quais se segue uma breve conclusão e recomendações para estudos posteriores.

[3] ver Eddelbuettel e François (2011)

2 REVISÃO BIBLIOGRÁFICA

2.1 Análise de Clusters

No atual cenário profícuo de desenvolvimento de ferramentas em *Machine Learning*, técnicas caras à Estatística têm sido constantemente revisitadas. Dentre estas está a análise de *clusters*.

Segundo Gan, Ma e Wu (2007, p. 5), a análise de *clusters* foi e tem sido exaustivamente explorada em todos os campos da Ciência, mas ainda carece de uma definição formal. Isto talvez aconteça, ao nosso ver, porque a definição que se tem como informal é bastante esclarecedora sobre o objetivo da técnica, qual seja: para um determinado conjunto de dados e para uma dada medida de dissimilaridade (ou similaridade), nós podemos reagrupar estas observações de modo que os pontos no mesmo grupo são similares e aqueles em grupos diferentes apresentam dissimilaridade (Gan, Ma e Wu, 2007, p.5). Muitos autores definem ainda as técnicas de *clustering* como técnicas de aprendizado não-supervisionado, uma vez que este tipo de algoritmo não necessita de qualquer informação prévia acerca dos grupos presentes nos dados para proceder ao agrupamento. Diferem, portanto, das técnicas de classificação, as quais denominam-se de aprendizagem supervisionada.

Muito embora a definição (informal) de *análise de clusters* faça menção a similaridade/dissimilaridade entre os pontos, existem tipos diferentes de agrupamentos que nos levam a assumir medidas também diferentes de similaridade para que seja possível identificar esses grupos nos dados. Nesse sentido, adotamos a ideia de Lorr (1983) *apud* Gan, Ma e Wu (2007, p. 6), o qual assevera que aparentam existir dois tipos de *clusters*: os compactos e os encadeados (*chained*). No primeiro tipo os pontos similares parecem se agrupar em torno de um centróide, como elipses, ao passo que no segundo, os pontos parecem seguir um caminho (*path*) formado pelas conexões dos pontos similares. As figuras 1 e 2 a seguir ajudam a ilustrar esta definição.

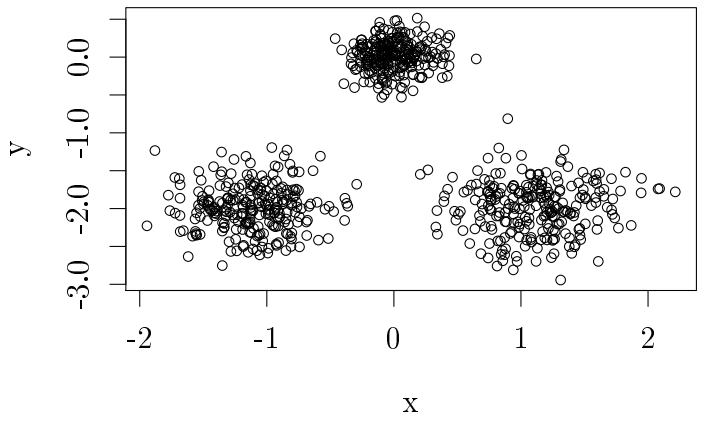

Figura 1 – *Clusters* Compactos

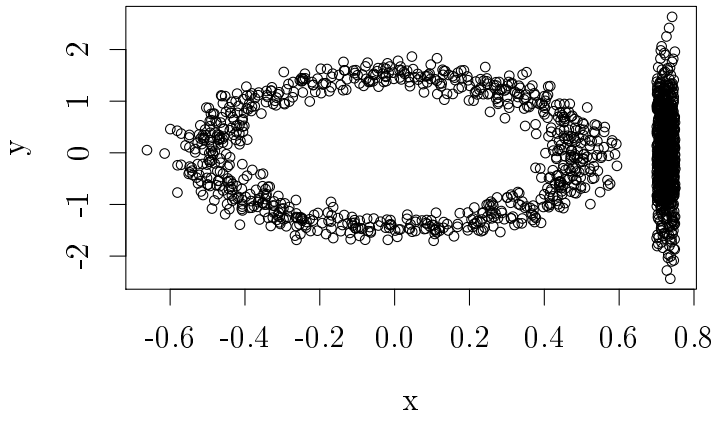

Figura 2 – *Clusters* encadeados

Como os agrupamentos podem se apresentar de diferentes formas nos dados, inclusive como uma mescla dos padrões apresentados nas figuras 1 e 2, inúmeros algoritmos de análise de *clusters* foram desenvolvidos com objetivo de reconhecer esses padrões nos dados. Uns são melhores para identificar os padrões apresentados na Figura 1; outros apresentam melhor performance no caso da Figura 2. Mesmo assim, dentre aqueles algoritmos que são mais adequados a um mesmo tipo de problema, existem diferenças quanto à abordagem

utilizada para atacar tal problema.

Procuraremos aqui traçar um quadro geral sobre os tipos de algoritmos de *clustering* existentes sem nos aprofundar muito acerca dos detalhes e funcionamento de cada um. Iniciamos pela figura 3 numa adaptação do que fora apresentado por Gan, Ma e Wu (2007, p. 10).

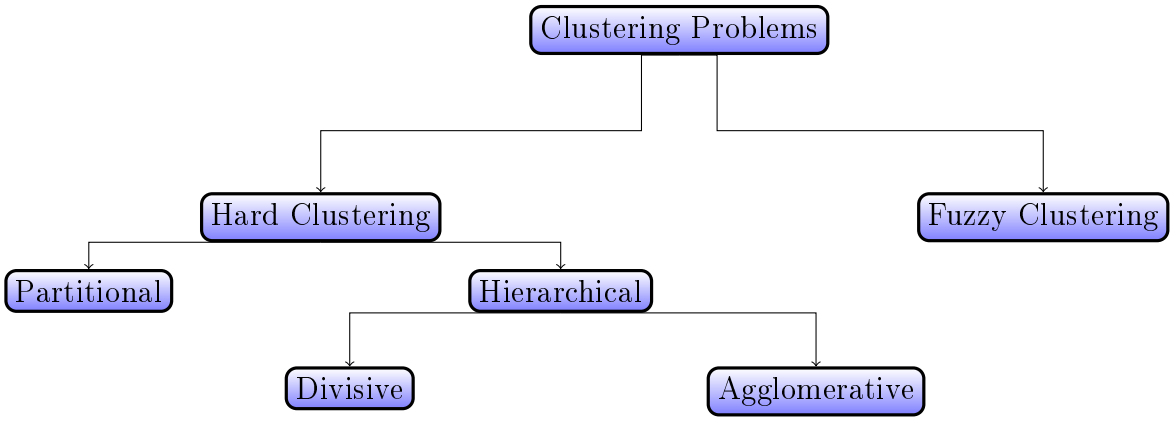

Figura 3 – Diagrama dos Algoritmos de Clusterização
Fonte: adaptada de Gan, Ma e Wu (2007, p. 10)

O primeiro nível de classificação separa aqueles algoritmos em que cada ponto só pode pertencer a um único grupo (*Hard Clustering*) daqueles em que os agrupamentos são feitos utilizando-se funções de pertinência que atribuem graus de pertencimento dos dados a cada grupo. Seguindo para o próximo nível do diagrama, os algoritmos particionais (não-hierárquicos) são aqueles em que os pontos são divididos em conjuntos que não se interseccionam, sendo que a qualidade desta divisão é avaliada por meio de uma função objetivo que convergirá para o melhor agrupamento possível dado o critério adotado pelo algoritmo. Já os métodos hierárquicos estabelecem um ramo de *clusters* para cada valor de k. São divisivos quando partem de uma configuração inicial de todas as observações em um único grupo e vão dividindo este grande grupo até chegar ao caso em que cada observação representa um único *cluster*. Os aglomerativos utilizam o procedimento inverso: começam com cada observação como um único grupo e vão aglomerando *clusters* até chegar à configuração com um único grupo.

Neste trabalho, embora o objetivo principal seja determinar o número de *clusters* nos dados, o método proposto depende da adoção de um algoritmo de agrupamento

a ser aplicado tanto aos dados originais quanto às simulações para posterior comparação e determinação de k. Por este motivo, em testes iniciais, utilizamos o *kmeans*, que é um dos algoritmos não-hierárquicos (partitional)[4] mais utilizados e já implementado em **R**. Contudo, os padrões de agrupamento que este tipo de algoritmo reconhece são limitados a elipsóides de tamanhos similares. Isso nos levou a adoção do algoritmo *MST based clustering* [5] a ser apresentado na seção 3.1.2.

2.2 Técnicas para determinação do número de *clusters*

As técnicas de análise de *clusters*, como já apontamos, utilizam uma série de suposições e variantes que as levam a ser mais uma técnica de análise exploratória de dados do que um fim em si mesma. Embora as soluções encontradas pelos algoritmos apresentados na seção anterior dependam de que o usuário/pesquisador informe o parâmetro k referente ao número de *clusters*, na maioria das vezes não se conhece o real número de grupos dos dados. Por isso, além dos algoritmos de análise de *clusters*, surgiram técnicas assessórias para se determinar um número k (ou testar a qualidade da solução para um dado k) a ser informado nos algoritmos de agrupamento. No entanto, da mesma forma que os algoritmos de clusterização, as técnicas de determinação de k, na maioria das vezes, também estarão sujeitas aos tipos de *clusters* (se são alongados, se são elipsoides, etc) e do próprio algoritmo de clusterização que estamos utilizando, o que faz disso um problema difícil de se resolver.

Gan, Ma e Wu (2007, p. 5) apresentam uma série de critérios para se avaliar as soluções retornadas pelos algoritmos de *clustering*. São os chamados *validity indices* que podem ser baseados em testes de hipótese, simulações, ou critérios não paramétricos [6]. No **R**, o pacote **NbClust** apresenta mais de 30 desses critérios que são utilizados em conjunto para se determinar o valor de k nos dados analisados. Para determinado conjunto de dados e para cada valor de k, calculam-se cerca de 30 índices, devendo-se adotar como k ideal aquele que aparece indicado como tal na maioria dos índices.

Na tentativa de incorporar um método de determinação do número de *clus-*

[4] ver Gan, Ma e Wu (2007)

[5] Gan, Ma e Wu (2007, p. 140) o classificam como algoritmo hierárquico, mas nossa implementação difere-se um pouco da comumente adotada e, por isso, discute-se este argumento na seção 3.1.2

[6] Pode-se utilizar ainda o próprio conhecimento do pesquisador sobre os dados para avaliar a qualidade da solução de um algoritmo de *clustering*.

ters a um algoritmo de agrupamento, Handl e Knowles (2007) construíram um algoritmo denominado MOCK *(multiobjective clustering with automatic k determination)*. Trata-se de uma abordagem do problema de aprendizagem não-supervisionada utilizando o *framework* da otimização multiobjetivo, que procura encontrar a melhor configuração dos dados utilizando vários critérios (concorrentes) e determinar automaticamente o k. A maioria dos algoritmos das categorias de análise de *cluster* que descrevemos na seção anterior utiliza um único critério para encontrar os grupos nos dados, dependendo do k informado. Os autores, então, reuniram critérios adotados em diferentes métodos de clusterização, juntamente com critérios das técnicas de validação (como Gap Statistic, CCC, etc) num único algoritmo de agrupamento.

Como os próprios autores demonstram, o algoritmo apresenta um excelente desempenho para vários tipos de *clusters*. Por outro lado, trata-se de um método relativamente complexo. Neste trabalho, nos inspiraremos nas ideias de Handl e Knowles (2007) a fim de desenvolver também um algoritmo para determinação do número de *clusters*, porém, mais simples e menos ambicioso. O objetivo é utilizar o *MST based clustering* como método de clusterização aplicado aos dados reais e aos dados simulados via hipercubos (CCC). Neste processo, serão coletadas 2 estatísticas de teste (conectividade e soma de quadrados intra-cluster), as quais serão utilizadas como parâmetros da otimização multiobjetivo.

Cabe destacar que, por se tratar de uma técnica de otimização, espera-se que o desempenho do algoritmo proposto seja bom em alguns casos e não tão bom em outros, em sintonia com o que advoga o *no free lunch (NFL) theorem*: " *equivalent performance of all optimization algorithms when averaged across all possible problems*" (Wolpert e Macready, 2006, p. 1). Portanto, não temos a ambição de que o método funcione para qualquer caso. Nosso objetivo é determinar justamente para quais situações, em particular, ele pode ser uma ferramenta útil para a identificação do número correto de *clusters* nos dados.

Nas seções que seguem, descreveremos de forma mais retida as técnicas e conceitos empregados em nosso algoritmo.

3 METODOLOGIA

3.1 Minimum Spanning Trees (MST)

Em Teoria dos Grafos, um grafo em que os vértices conectam-se sem que haja ciclos é definido como uma árvore Chartrand (1975, cap. 4). Por conseguinte, uma *spanning tree* ou árvore geradora é qualquer árvore ligando todos os vértices deste grafo. Se esta árvore for a menor árvore que liga os vértices do grafo, temos, então, uma árvore geradora mínima ou *minimum spanning tree*. Quando dizemos menor árvore, queremos dizer com o menor custo possível, supondo que as arestas do grafo possuam custos associados. Todos estes conceitos são ilustrados no conjunto de figuras abaixo:

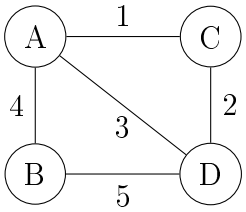

Figura 4 – Grafo não-direcional

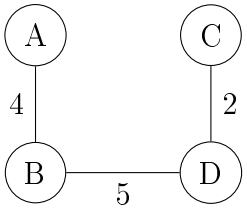

Figura 5 – Sapanning Tree qualquer

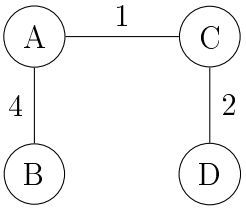

Figura 6 – Minimum Spanning Tree

Note na Figura 5 que esta árvore possui custo total dado pela soma dos pesos das arestas, $4+5+2=11$, ao passo que na Figura 6 o custo total é $4+1+2=7$, sendo esta portanto, a *Minimum Spanning Tree (MST)*. A definição pormenorizada de grafos e árvores não estão no escopo desse trabalho. No entanto, o conceito de *Minimum Spanning Trees - MST)* será tratado em maior profundidade.

Pop (2002, p. 3) apresenta uma definição formal de *MST*, a qual adotamos também neste trabalho:

Seja $G = (V, E)$ um grafo conectado que possui uma função positiva de custo $c \geq 0$ definida no conjunto de vértices E.

Definição 3.1: *Minimum Spanning Tree*

Uma árvore geradora mínima ou minimum spanning tree (MST) de G é uma árvore geradora T^* de G que possui custo mínimo, ou seja,

$$c(E(T^*)) = min\{c(E(T))|T \text{ é árvore geradora de } G\} \quad (1)$$

Vale ressaltar que, embora as definições tradicionais de *MST* sejam baseadas em grafos, é mais comum, quando lidamos com dados, trabalhar com o formato de matriz de distâncias. Ambos os formatos são equivalentes, tratando-se apenas de uma conveniência relativa ao formato dos dados de entrada [7]. No caso da matriz de distâncias, os custos associados às arestas equivalem às distâncias Euclidianas entre os pontos. Portanto, o problema de *MST* resume-se a encontrar os vizinhos mais próximos para cada ponto.

A distância Euclidiana entre os pontos **p** e **q** é o comprimento do segmento de linha que os conectam ($\overline{\mathbf{pq}}$).

Definição 3.2: *Distância Euclidiana*

Em coordenadas cartesianas, se $\boldsymbol{p} = (p_1, p_2, ..., p_n)$ *e* $\boldsymbol{q} = (q_1, q_2, ..., q_n)$ *são dois pontos no espaço Euclidiano de ordem n, então a distância (d) de* $\boldsymbol{p}$ *até* $\boldsymbol{q}$*, ou de* $\boldsymbol{q}$ *até* $\boldsymbol{p}$ *é dada pela fórmula de Pitágoras:*

$$d(\boldsymbol{p},\boldsymbol{q}) = d(\boldsymbol{q},\boldsymbol{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + ... + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \quad (2)$$

3.1.1 Dual Tree Boruvka MST

Historicamente, o problema de árvores geradoras mínimas surge com Boruvka na década de 1920, quando este estudava uma forma de estabelecer a malha elétrica da Morávia da maneira mais eficiente possível. Por ser naturalmente um problema de otimização, a partir de Boruvka, vislumbraram-se várias aplicações para a teoria de *MST*, como por exemplo em redes de transporte, redes de comunicação, conexões cabeadas, dentre outras (Jayawant e Glavin, 2009, p.439; Graham e Hell, 1985, p.43).

[7] este foi um dos motivos de não adotarmos pacotes disponíveis no R

Após seu advento, muito autores contribuiram com algoritmos para o problema de *MST*. Dentre os algoritmos mais utilizados até os dias atuais estão o de *R. C. Prim* e o de *J. B. Kruskal*, que foram desenvolvidos na década de 1950[8](Jayawant e Glavin, 2009, p.439).

Neste trabalho, inicialmente optamos pela implementação do algoritmo de Prim em **C++** que seria chamado por uma função do **R** via pacote **Rcpp**. Contudo, devido ao nosso algoritmo ser intensivo computacionalmente e devido ao fato de nossa principal preocupação inicial não ter sido a eficiência computacional, dados um pouco acima de milhares de observações resultavam em um processamento bastante lento, mesmo com a utilização de um API *(Application Programming Interface)* entre **R** e uma linguagem rápida como **C++**.

Após algumas pesquisas, resolvemos adotar o algoritmo de *MST* presente na biblioteca de *Machine Learning* de **C++** chamada **mlpack**(Curtin et. al., 2011)[9].

Segundo do site que hospeda a documentação sobre a biblioteca,

" ***mlpack*** *is a C++ machine learning library with emphasis on scalability, speed, and ease-of-use. Its aim is to make machine learning possible for novice users by means of a simple, consistent API, while simultaneously exploiting C++ language features to provide maximum performance and maximum flexibility for expert users. This is done by providing a set of command-line executables which can be used as black boxes, and a modular C++ API for expert users and researchers to easily make changes to the internals of the algorithms.*" (http://www.mlpack.org/about.html)

Trata-se de uma biblioteca com implementações extremamente eficientes se em comparação com os *toolkits* concorrentes na área de aprendizagem de máquinas, como MATLAB e *scikitlearn* da linguagem Python (Curtin et. al., 2011, p.4). Dentre os algoritmos implementados nesta biblioteca está o *Dual Tree Boruvka*, o qual utilizamos para encontrar as árvores geradoras mínimas em nossos dados.

O *Dual Tree Boruvka* consiste em uma modificação do algoritmo original de Boruvka, utilizando a técnica de *Dual Trees*. Segundo March, Ram e Gray (2010, p. 606), algoritmos do tipo *Dual Tree* são um *framework* computacional bastante utilizado para solução de problemas físicos, estatísticos e de aprendizagem de máquina, devido ao seu desempenho

[8]Na verdade, o algoritmo conhecido como algoritmo de Prim fora desenvolvido ainda na década de 1930, por V. Jarník (Jayawant e Glavin, 2009, p.439; Graham e Hell, 1985, p.43))

[9]ver http://www.mlpack.org/

consideravelmente mais rápido se em comparação a outros algoritmos tradicionais.

Em um problema de *MST*, os algoritmos normalmente executam uma busca pelos vizinhos mais próximos baseando-se na distância Euclidiana como já mencionamos. Isso faz com que o tempo de execução do algoritmo inicialmente proposto por Boruvka assim como o dos algoritmos de Prim e Kruskal dependam do método utilizado para encontrar o par mais próximo de cada ponto. A diferença para um algoritmo do tipo *Dual Tree* está no fato deste calcular simultaneamente todos os vizinhos mais próximos, reduzindo o número de computações entre pontos diferentes[10].

Com a utilização deste algoritmo para obtenção da árvore geradora mínima, nosso método passou a ser mais eficiente e confiável em termos computacionais devido ao fato de **mlpack** ser uma biblioteca desenvolvida por especialistas e de manutenção contínua. Para carregar as funções da biblioteca **mlpack** de dentro do **R**, confeccionou-se um *script* em **C++**, utilizando-se API's fornecidos pelos pacotes **Rcpp** e **RcppMLPACK**.

Dessa forma, temos a base de nosso algoritmo para detecção do número de clusters: a obtenção inicialmente de uma árvore geradora mínima dos dados. A partir desta árvore, utilizaremos métodos para particionarmos os dados em diferentes *clusters*. Os métodos de clusterização que utilizam como base árvores geradoras mínimas são denominados *MST based clustering*.

3.1.2 MST based clustering

Como já dissemos antes, no nosso caso, os custos da árvore geradora mínima são as distâncias Euclidianas entre os pontos presentes em nossos dados. A transição de um algoritmo apenas de *MST* para um *MST based clustering* dá-se, *grosso modo*, quando passamos a identificar, dentro da árvore geradora mínima, arestas que são muito longas em comparação com o restante da árvore. Ao eliminarmos essas ligações, passamos a ter, então, os *clusters*, conforme se observa nas figuras de 7 a 9:

[10]para informações mais detalhadas sobre o funcionamento do algoritmo, ver March, Ram e Gray (2010)

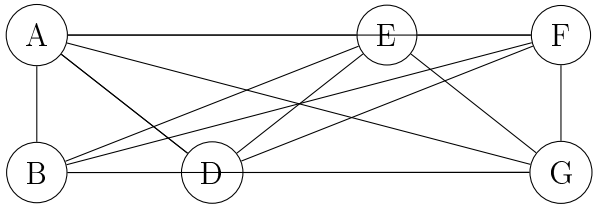

Figura 7 – Representação da matriz de distâncias dos dados via Gráfo

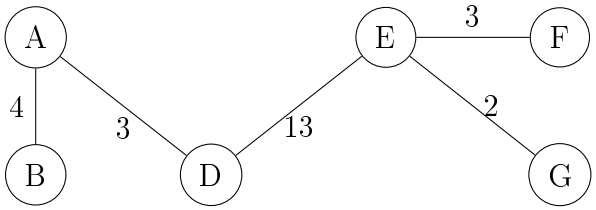

Figura 8 – Minimum Spanning Tree resultante

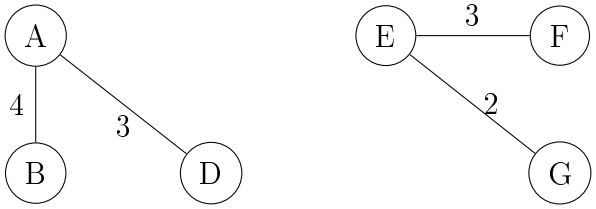

Figura 9 – MST based clusters

Na Figura 9, fora eliminada a ligação entre os pontos D e E, que era a aresta de maior distância dentre todas as ligações restantes da árvore geradora mínima. Em nosso algoritmo, no entanto, o critério de eliminação é um pouco mais complexo. Ao invés de considerarmos apenas as distâncias, trabalharemos com o conceito de links interessantes (Handl e Knowles, 2007), para conseguirmos identificar possíveis outliers nos dados e não classificá-los sempre como um *cluster* separado. O conceito de link interessante será explicado na próxima seção em que descreveremos os procedimentos de forma completa. O exemplo anterior serve apenas para ilustrar a ideia geral acerca da formação de *clusters* a partir de uma árvore geradora mínima.

Cabe, neste ponto, uma nota acerca da diferença entre o método hierárquico de clusterização conhecido como *single linkage cluster* e os métodos *MST based clustering*.

Embora muitos autores afirmem que os métodos se equivalem, vale ressaltar que os algoritmos baseados em *MST* conectam os pontos, ao passo que o método *single linkage* conecta hierarquicamente partições dos dados e não os pontos individualmente (Oksanen, 2014, p.7), o que resulta em soluções diferentes para os mesmos dados.

3.1.3 MST based clustering adotado no trabalho

Para este trabalho, implementamos um método particular para clusterização dos dados baseado em *MST*, utilizando o conceito de links interessantes apresentado por Handl e Knowles (2007). Em termos gerais, nossa implementação de *MST based clustering* recebe como parâmetros de entrada o número de clusters k a ser formado e o número de vizinhos mais próximos n_neig a ser considerado para a identificação de links interessantes.

Um link interessante é definido como a ligação entre um ponto i e um ponto j ($i \rightarrow j$) em que j não está entre os n_neig vizinhos mais próximos de i, assim como i não está entre os n_neig vizinhos mais próximos de j. Este link é, então, eliminado. Repete-se esse procedimento até que sejam eliminadas $k-1$ ligações. Caso o número de links interessantes seja maior que $k-1$, eliminam-se aqueles links de maior distância euclidiana $d(i,j)$. Caso o número de links interessantes seja menor que $k-1$, eliminam-se as maiores arestas (mesmo não sendo links interessantes) até que se alcance $k-1$ eliminações e consequentemente k *clusters* nos dados.

Suponha que desejamos rodar nosso *MST based clustering* para $k = 2$. A Figura 10 indica a árvore geradora mínima encontrada.

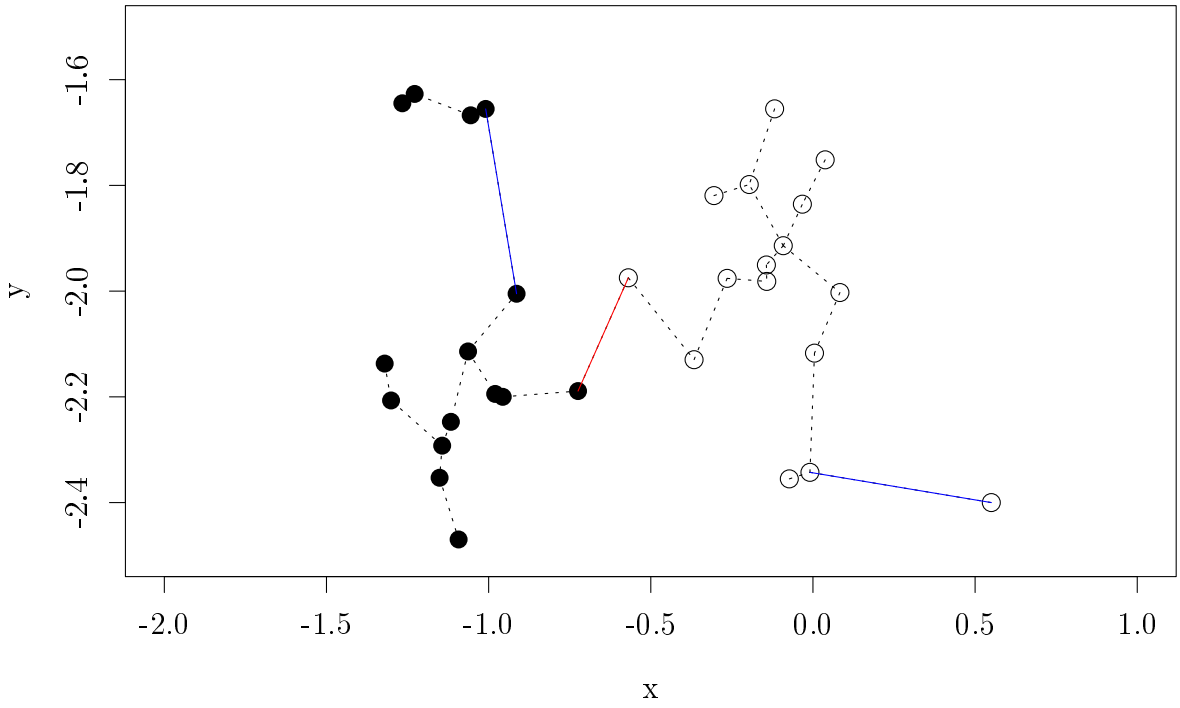

Figura 10 – Identificação de links interessantes

Para obtermos dois *clusters* a partir da *MST* da Figura 10, precisamos eliminar $k-1$ arestas, ou seja, $2-1=1$ aresta. Suponha que estabelecendo um critério de $n\_neig=5$ vizinhos mais próximos, a aresta marcada em vermelho indica a aresta a ser eliminada ao se adotar o critério de link interessante: o ponto à direita da aresta não se encontra entre os 5 vizinhos mais próximos do ponto à esquerda. Se utilizássemos apenas um critério de distância, teríamos duas arestas marcadas em azul que são maiores que a vermelha, mas que indicam, na verdade, outliers em nossos dados. A eliminação de qualquer uma das arestas azuis ao invés da vermelha não geraria a configuração de grupos condizente com a real divisão existente (pontos pretos vs pontos brancos).

Uma vez definido o agrupamento dos pontos para um dado k, temos de calcular as estatísticas de teste que servirão ao algoritmo mais geral de determinação do número adequado de *clusters*. Como trabalharemos com a otimização de uma função multiobjetivo, as estatísticas de teste a serem calculadas são a soma de quadrados intra-cluster e a conecti-

vidade.

3.1.3.1 Soma de quadrados intra-cluster

Seja $\mathbf{D}_s$ a matriz $N \times N$ dos **quadrados** das distâncias euclidianas entre os pontos; $\mathbf{g}$ um vetor coluna de tamanho N contendo a sequência de $1, 2, 3, ..., k$ rótulos dos *clusters*; e $\mathbf{G}$ a *design matrix* (matriz com zeros e uns indicando o pertencimento a determinado cluster - também chamada de *model matrix*) confecionada a partir do vetor $\mathbf{g}$: $\mathbf{G} = \textit{design(}\boldsymbol{g}\textit{)}$. Podemos calcular a soma de quadrados total (SSt) da seguinte forma:

$$SSt = \frac{\sum_i^N \sum_j^N d_{ij}}{2N}, \tag{3}$$

onde

- d_{ij} indica o elemento da linha i e coluna j da matriz $\mathbf{D}_s$;

Se $\mathbf{n}$ é o vetor contendo os inversos dos totais das colunas da matriz $\mathbf{G}$, o vetor $\mathbf{SSw}$ contendo a soma de quadrados dentro de cada *cluster* pode ser obtido via:

$$\mathbf{SSw} = \text{diag}\left(\frac{\mathbf{G'} * \mathbf{D}_s * \mathbf{G}}{2} * \mathbf{n'}\right) \tag{4}$$

Por conseguinte, nossa estatística de teste *soma de quadrados total intra-cluster* ($SS.wt$) é calculada somando-se os elementos deste útlimo vetor:

$$SS.wt = \sum_{i=1}^{k} SSw_i \tag{5}$$

3.1.3.2 Conectividade

O índice de conectividade, por sua vez, é calculado comparando-se cada ponto aos seus vizinhos mais próximos. Caso o $j-$ésimo vizinho mais próximo esteja no mesmo *cluster* que o ponto avaliado, não é atribuída qualquer penalidade. Do contrário, caso o $j-$ésimo vizinho mais próximo não esteja no mesmo *cluster*, é inputada uma penalidade da ordem $1/j$, a qual é gradualmente decrescente conforme se avança na avaliação dos vizinhos

mais próximos. Este procedimento é repetido até que todos os pontos sejam percorridos, somando-se as penalidades para a solução com número de *clusters* k. O objetivo é minimizar o nível de conectividade.

Para uma definição mais formal do cálculo da conectividade, adotaremos a utilizada por Handl e Knowles (2007, p. 60).

Seja N o tamanho ou número de pontos no *dataset*; L (ou n_neig) a cardinalidade do conjunto de vizinhos mais próximos que adotamos para avaliar a contribuição para a conectivade ; $\mathbf{W}$ a matriz de ordem $L \times N$ de distâncias euclidianas ordenada ao longo das linhas; e C_k o *cluster* de rótulo k. Podemos calcular a conectividade da seguinte forma:

$$conn = \sum_{i=1}^{N} \left(\sum_{j=1}^{L} x_{j,i} \right), \tag{6}$$

$$onde\ x_{j,i} = \begin{cases} \frac{1}{j}, & \text{se } \nexists C_k : i \in C_k \text{ e } j \in C_k. \\ 0, & \text{caso contrário.} \end{cases}$$

sendo j o *j-ésimo* vizinho mais próximo do ponto representado pela coluna i da matriz $\mathbf{W}$;

Agora que finalizamos a apresentação dos aspectos básicos relacionados à parte de clusterização de nosso algoritmo, partiremos para a apresentação metodológica do conceito de *Cubic Clustering Criterion (CCC)* a ser utilizado na confecção de simulações em nossa proposta.

3.2 Cubic Clustering Criterion (CCC)

Segundo Sarle (1983, p. 2), o problema mais difícil na análise de *clusters* é como estimar o número de grupos presente nos dados, corroborando nossos apontamentos na introdução deste trabalho. O autor propõe, então, um método para tal finalidade, utilizando uma estrutura de teste de hipóteses onde:

H 0: *os dados foram amostrados de uma distribuição Uniforme em um hipercubo;*

H 1: *os dados foram amostrados de uma mistura de distribuições normais multivariadas esféricas com variâncias iguais e probabilidades de amostragem iguais.*

Para gerar o hipercubo, são obtidas as direções de maior variabilidade dos dados via *Principal Component Analysis - PCA* e identificados os respectivos valores máximo e mínimo em cada dimensão transformada. Com esses parâmetros, então, geram-se Uniformes(min_j, max_j) nas j dimensões, conforme ilustrado na Figura 11 com dados em $\mathbb{R}^2$.

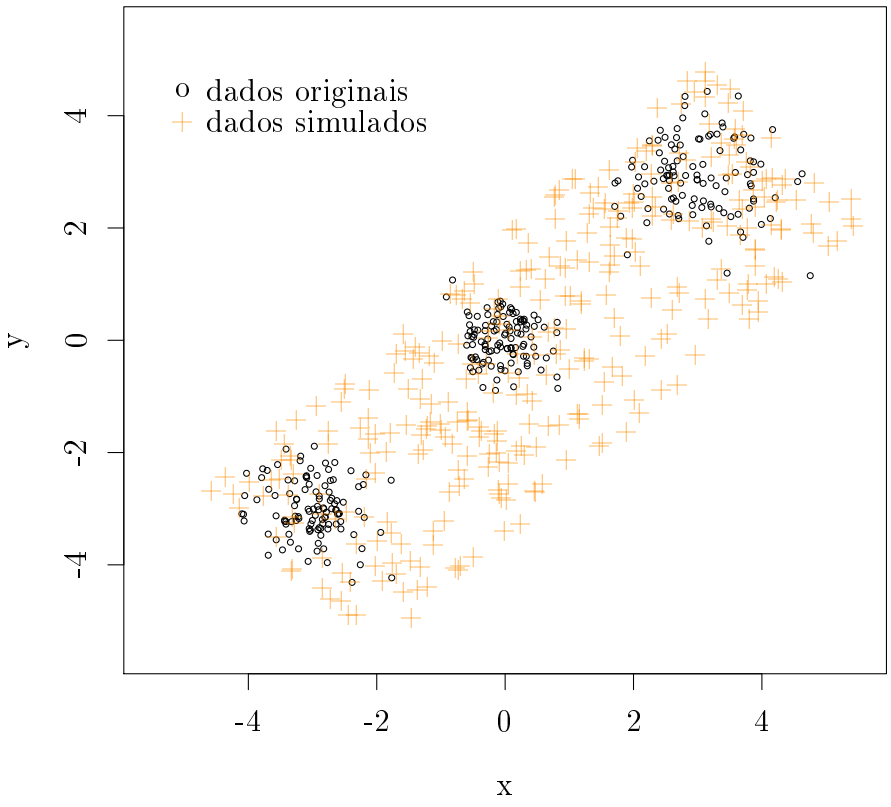

Figura 11 – Hipercubo (ou quadrado) em $\mathbb{R}^2$

Sarle (1983), em seu método, obteve uma aproximação para a distribuição do coeficiente de determinação R^2 sob a hipótese nula, qual seja, de que os dados provêm de uma distribuição Uniforme em um hipercubo (um parelelepípedo de dimensão p). O R^2, neste caso, é entendido como a proporção de variância com a qual cada *cluster* contribui para a variância total dos dados (soma de quadrados total). O *cubic clustering criterion CCC* é

então obtido ao se comparar o R^2 observado nos dados após clusterização com o $E(R^2)$ sob a hipótese nula, utilizando uma transformação para estabilização da variância aproximada.

Valores positivos do CCC significam que o R^2 dos dados é maior do que se esperaria de uma amostra de distribuição Uniforme, indicando portanto a presença de *clusters*. Dessa forma, para $k = 1, 2, 3, ..., n$, o maior valor de CCC indica o número de clusters existente nos dados [11].

Conforme apontamos, o método é executado após a clusterização dos dados. Para esta finalidade (e também para obtenção da distribuição de referência de R^2), o autor utiliza (utilizou) o método de mínima variância de Ward (Sarle, 1983, p.8).

Em nosso método, não utilizaremos o índice CCC em si[12], mas apenas a ideia de gerar hipercubos a partir de distribuições Uniformes como uma distribuição de referência para os dados. Espera-se que configurações equivocadas de *clusters* para um determinado k nos dados apresentem estatísticas de teste próximas àquelas observadas no hipercubo para o mesmo valor de k, indicando que a divisão encontrada nos dados ocorre por mero acaso. Por outro lado, quando estivermos comparando as estatísticas de teste dos dados para o k ideal, espera-se que estas fiquem mais afastadas das estatísticas do hipercubo para o mesmo k.

3.3 Dominância e Otimalidade de Pareto

Ao utilizarmos duas estatísticas de teste, precisaremos de um método para indicar soluções ótimas a partir de vários pares desses valores. Para este fim, adotamos os conceitos de dominância e otimalidade de Pareto. Estes conceitos são, na verdade, sinônimos e se relacionam à otimização da função multiobjetivo. Otimizar esta função consiste justamente na otimização de quantidades concorrentes.

A diferença entre algoritmos que utilizam uma função de otimização uniobjetivo e aqueles que utilizam uma função multiobjetivo reside no número de soluções ótimas resultantes nos dois casos. Enquanto que no problema de otimização uniobjetivo haverá uma única solução, na abordagem multiobjetivo poderá haver mais de uma solução igualmente ótima.

[11] No **R**, uma implementação do CCC está presente no pacote **NbClust**

[12] Para maiores detalhes acerca da forma como se calcula o índice CCC, sugere-se a consulta ao relatório técnico de Sarle(1983).

Conforme apontam Mishra e Harit (2010, p. 35), a maioria dos algoritmos de otimização multiobjetivo utilizam o conceito de dominância na sua busca por soluções. No algoritmo proposto neste trabalho buscamos pontos não-dominados nos conjuntos formados pelas estatísticas de teste de conectividade e soma de quadrados intra-cluster tanto nos dados reais quanto nos dados simulados. Estas estatísticas são funções de k com comportamentos concorrentes: quanto maior o número de clusters, maior é a conectividade calculada e menor é a soma de quadrados intra-cluster.

Segundo os mesmos autores, as definições de pontos não-dominados e de conjunto de pontos não-dominados são as seguintes:

Definição 3.3: *(dominância) A solução* $\mathbf{x}^{(1)}$ *domina a solução* $\mathbf{x}^{(2)}$ *se ambas as condições* 1 *e* 2 *abaixo são verdadeiras:*

1. *A solução* $\mathbf{x}^{(1)}$ *não é pior que* $\mathbf{x}^{(2)}$ *em nenhum dos objetivos, ou* $f_j(\mathbf{x}^{(1)}) \not\triangleright f_j(\mathbf{x}^{(2)})$ $\forall j = 1, 2, ..., M$*;*

2. *A solução* $\mathbf{x}^{(1)}$ *é estritamente melhor que* $\mathbf{x}^{(2)}$ *em pelo menos um objetivo, ou* $f_j(\mathbf{x}^{(1)}) \triangleleft f_j(\mathbf{x}^{(2)})$ *para um* $j \in \{1, 2, ..., M\}$*;*

Definição 3.4: *(conjunto de pontos não-dominados) Dentre um conjunto de soluções* P*, o conjunto não-dominado de soluções* P^* *é aquele que não é dominado por nenhum membro do conjunto* P*.*

A figura 12 a seguir ilustra a identificação de pontos não dominados e estabelece o que se chama de fronteira de Pareto em um conjunto de dados referentes a estatísticas de testes calculadas durante uma execução de nosso algoritmo.

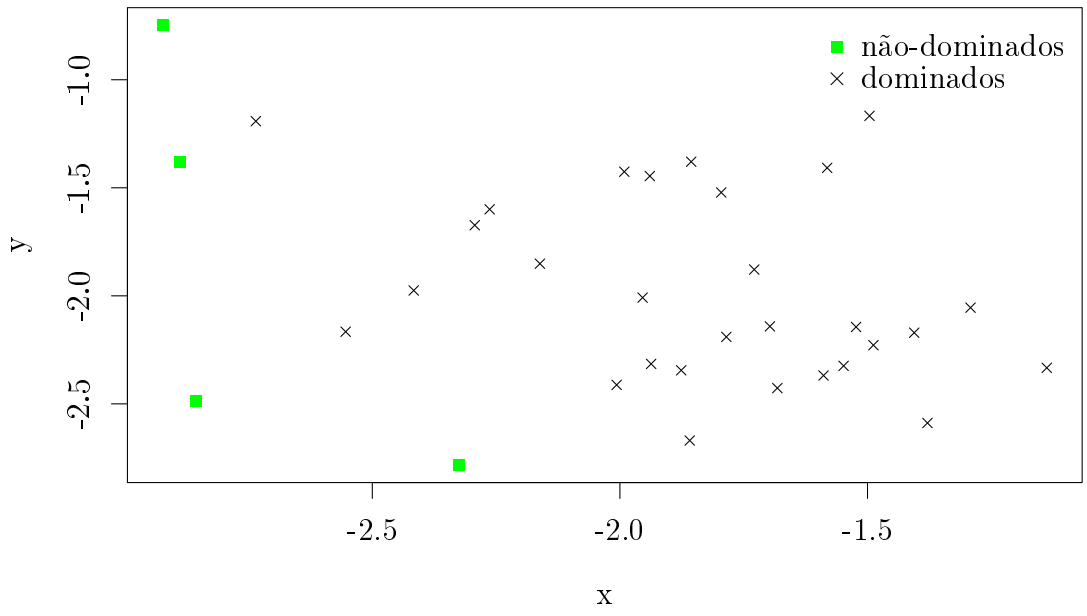

Figura 12 – Pontos não-dominados (Otimalidade de Pareto)

Na nuvem de pontos representada acima, apenas aqueles marcados em verde atendem às condições de não-dominância descritas anteriormente.

3.4 Descrição do método proposto para determinação do número de clusters

Em linhas gerais, o método proposto neste trabalho para a detecção do número *clusters* consiste em aplicar um algoritmo de clusterização nos dados em análise para $k = 2, 3, 4, 5, ..., n$ grupos, armazenando-se as estatísticas de soma de quadrados total intra-cluster e conectividade resultantes de cada configuração. Esse procedimento é, em seguida, repetido para um conjunto de simulações.

Ao final do processo, para cada k, comparam-se as estatísticas de teste não-dominadas dos dados reais e a nuvem de pontos não-dominados das estatísticas de teste resultantes das simulações. Aquele ponto não-dominado das estatísticas de testes dos dados que apresentar a maior distância para o centróide da nuvem de pontos não-dominados (de mesmo k) das estatísticas de teste oriundas das simulações será o número de *clusters*

adotado[13].

A particularidade do método está na adoção de uma função multiobjetivo de estatísticas de teste e também no modo como os dados simulados são gerados. Na etapa de simulação é onde se utiliza a ideia de *Cubic Clustering Criterion (CCC)*. Conforme já apresentamos na seção 3.2, após capturarmos as direções de maior variabilidade dos dados via *Principal Component Analysis - PCA*, obtemos o máximo e mínimo dos dados em cada dimensão transformada. De posse desses parâmetros, geram-se distribuições Uniformes(min_j, max_j) em cada uma das j dimensões, formando o que denominamos "hipercubo".

Nas simulações, são gerados hipercubos em um número definido pelo usuário. Sobre cada elemento dessa lista de hipercubos (ou dados simulados) é aplicado o mesmo algoritmo de clusterização utilizado nos dados originais. Coletam-se novamente as duas estatísticas de teste e, para cada k, selecionam-se aqueles pontos que não são dominados pelos demais.

A proposta inicial do método era realizar testes com diversas combinações de técnicas de clusterização. No entanto, apenas testes com a aplicação conjunta de *MST based clustering* e do algoritmo *kmeans* foram feitos. Como os resultados com *kmeans* não foram diferentes daqueles utilizando somente *MST*, optou-se por utilizar apenas o algoritmo *MST based clustering* apresentado na seção 3.1.2.

Um dos problemas com o *kmeans* seria sua instabilidade na geração de resultados. Dependendo dos pontos que o algoritmo adota para inicialização, a configuração de *clusters* resultante e consequentemente as estatísticas de teste serão bastante diferentes. Além disso, densidades de pontos diferentes em cada *cluster* também acabam por distorcer os resultados com o *kmeans* [14]. Restrições de tempo não nos permitiram testar outros algoritmos de clusterização, nem outras estatísticas de teste.

3.4.1 Algoritmo para determinação do número de clusters

Apresentadas as metodologias e os principais conceitos utilizados no desenvolvimento do método proposto, passamos a detalhar o algoritmo geral de determinação do

[13] a distância das soluções não-dominadas dos dados para o centróide da nuvem de soluções não-dominadas das simulações é ponderada/dividida pela variabilidade desta nuvem de pontos

[14] ver http://varianceexplained.org/r/kmeans-free-lunch/

número de *clusters*. Para isso, o dividimos em quatro etapas com subalgoritmos. Algumas etapas são acompanhadas de uma sequência de figuras para dados no $\mathbb{R}^3$ que servem para ilustrar os procedimentos adotados, ligando-os aos conceitos e metodologia já apresentados nas seções anteriores.

- **Etapa 1:**

Algoritmo 1: Clusterização dos dados originais via MST para $k = 2, 3, ..., n$

Data: Dados originais

Input: $kmax$ = nro máximo de clusters a se testar, n_neig = número de vizinhos mais próximos

Result: Dados originais agrupados e respectivas estatísticas de teste para cada solução $k = 2, 3, ..., n$

```
inicialização;
calcular a arvore geradora mínima;
identificar os n_intlink links interessantes;
if <n_intlink == k − 1> then
    eliminar única aresta de link interessante existente;
else if <n_intlink > k − 1> then
    eliminar links interessantes até cortar k − 1 arestas em ordem
     decrescente de distância;
else
    eliminar todos os links interessantes;
    eliminar as maiores arestas que não são links interessantes até obter
     k − 1 eliminações;
end
calcular estatísticas de teste (soma de quadrados intra-cluster
 SS.wt.obs e conectividade conn.obs);
```

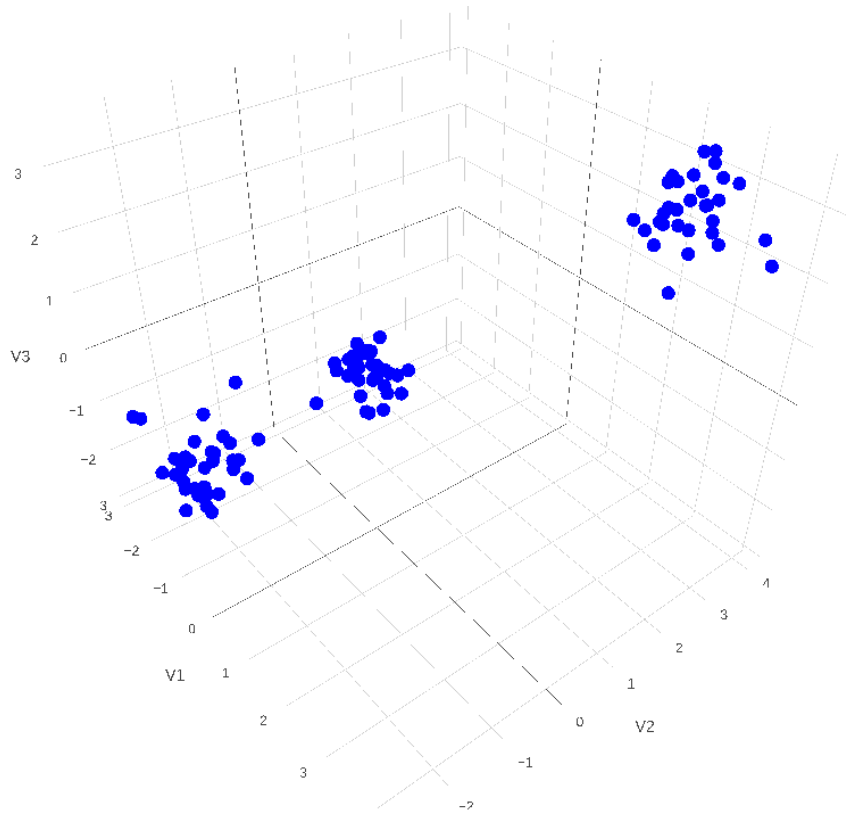

Figura 13 – Dados

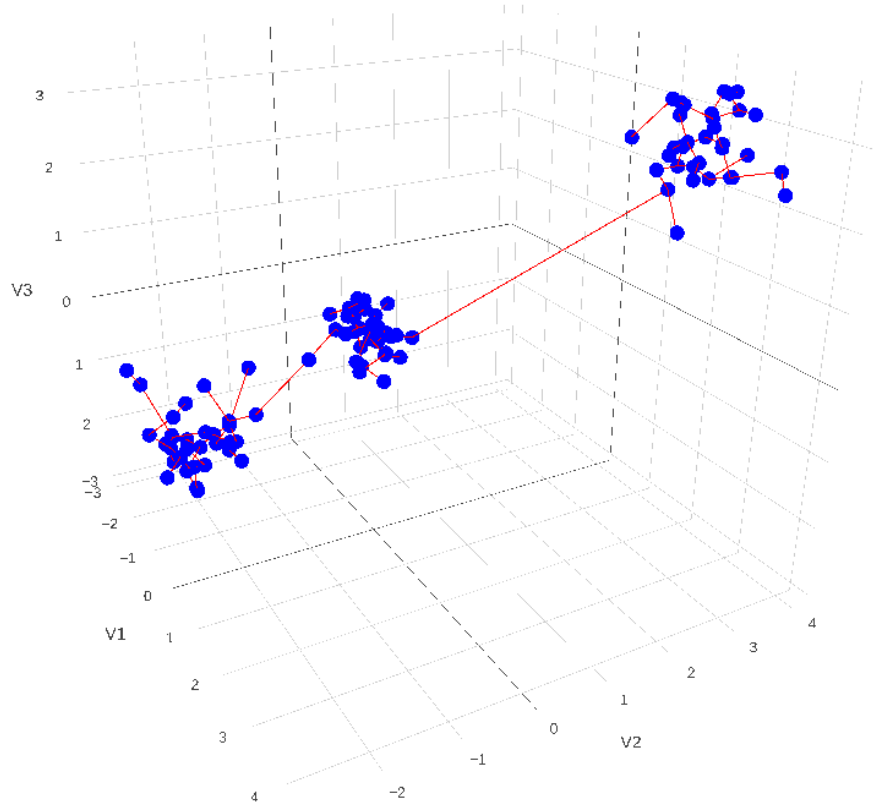

Figura 14 – MST

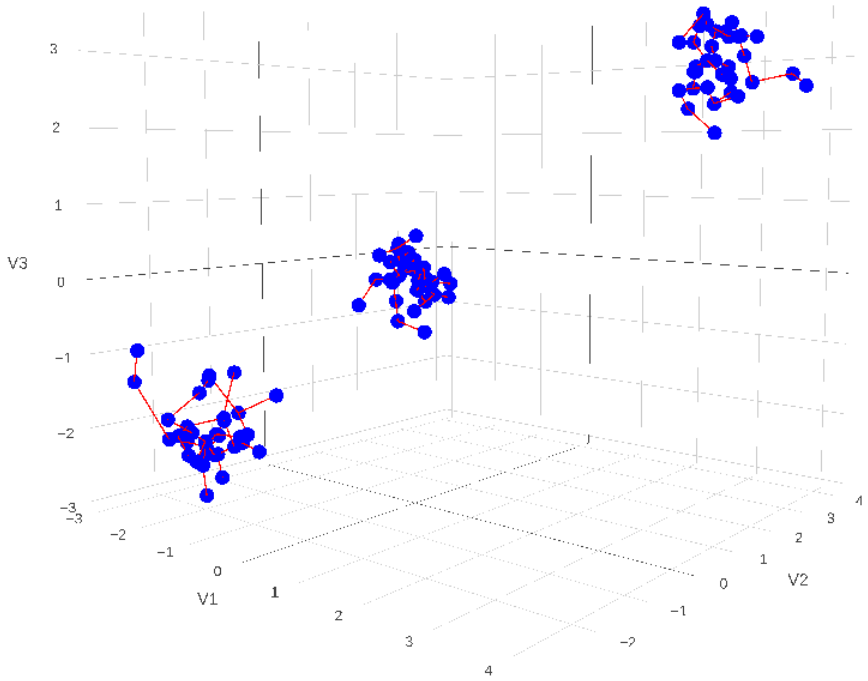

Figura 15 – MST based clustering

- **Etapa 2:**

Algoritmo 2: Construção dos hipercubos - simulações dos dados

Data: Dados originais

Input: min_j, max_j, $n.sim = 1000$

Result: Lista de hipercubos

1 inicialização;

2 realizar *Principal Component Analysis* - *PCA* nos dados originais;

3 colocar os dados originais no espaço transformado;

4 obter extremos de cada dimensão transformada j dos dados originais (min_j, max_j);

5 gerar $n.sim$ distribuições Uniformes $U(min_j, max_j)$ (hipercubos), armazenando-as em uma lista;

6 colocar os dados simulados no espaço original;

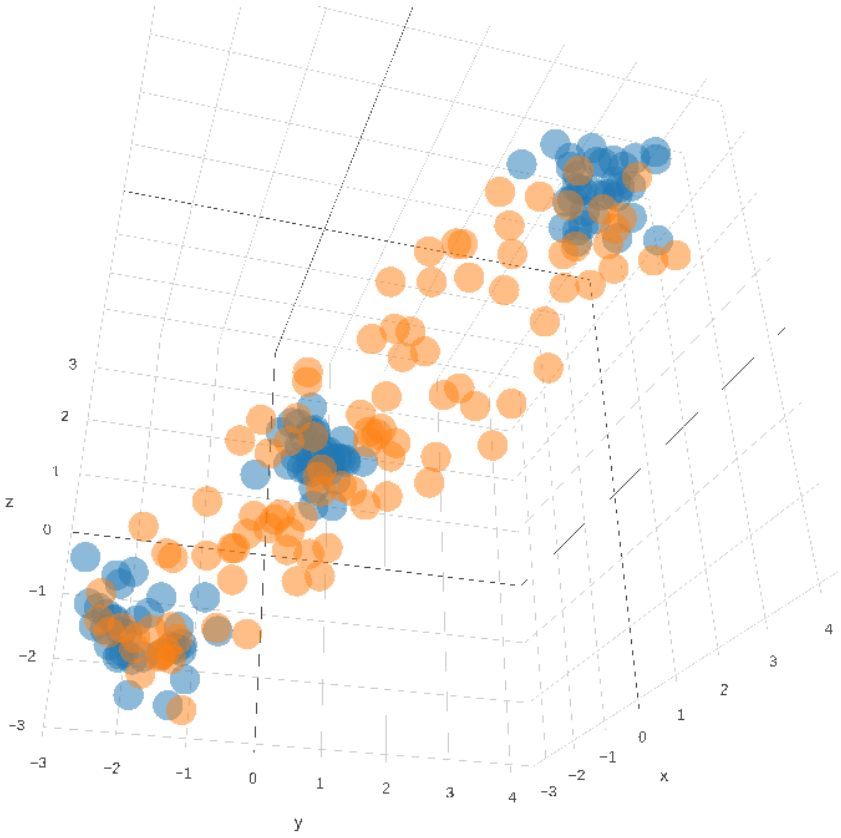

Figura 16 – Um hipercubo dos dados simulados

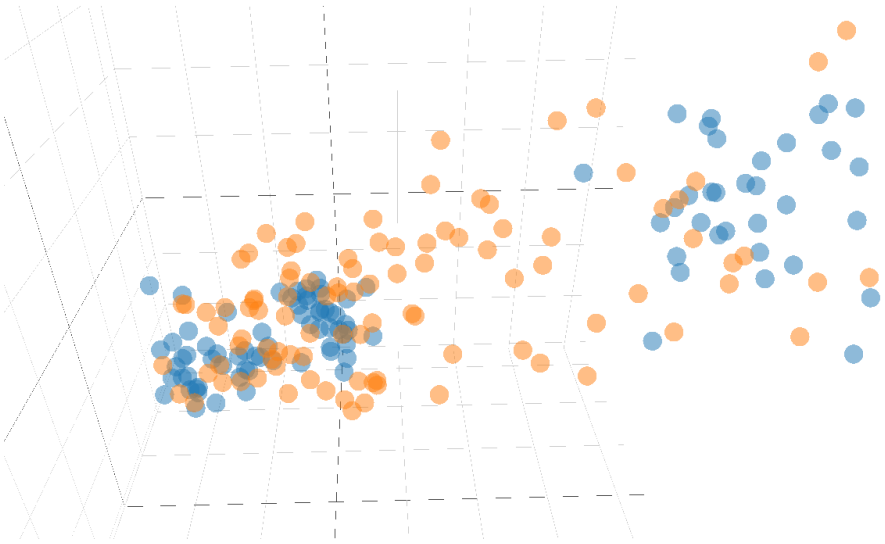

Figura 17 – Um hipercubo dos dados simulados (em perspectiva aproximada)

- **Etapa 3:**

Algoritmo 3: Clusterização dos dados simulados via MST para $k = 2, 3, ..., n$

Data: Dados simulados

Input: $kmax$ = nro máximo de clusters a se testar, n_neig = número de vizinhos mais próximos

Result: Dados simulados agrupados e respectivas estatísticas de teste para cada solução $k = 2, 3, ..., n$

```
1  inicialização;
2  calcular a arvore geradora mínima;
3  identificar os n_intlinklinks interessantes;
4  if <n_intlink == k − 1> then
5  |  eliminar única aresta de link interessante existente;
6  else if <n_intlink > k − 1> then
7  |  eliminar links interessantes até cortar k − 1 arestas em ordem
   |   decrescente de distância;
8  else
9  |  eliminar todos os links interessantes;
10 |  eliminar as maiores arestas que não são links interessantes até obter
   |   k − 1 eliminações;
11 end
12 calcular estatísticas de teste (soma de quadrados intra-cluster
     SS.wt.obs e conectividade conn.obs);
```

- **Etapa 4:**

Algoritmo 4: Determinação do k número de clusters

Data: Dados originais agrupados; Dados simulados agrupados

Input: estatísticas de teste: $SSwt.obs$, $conn.obs$; $SSwt.sim$, $conn.sim$

Result: k escolhido

1 inicialização;

2 identificar os pontos não dominados para as estatísticas de teste das soluções relativas aos dados originais: $non.dom.obs_k$;

3 identificar os pontos não dominados para as estatísticas de teste das soluções relativas aos dados simulados: $non.dom.sim_k$;

4 para cada k, calcular as distâncias euclidianas entre $non.dom.obs_k$ e o centroide da nuvem de pontos $non.dom.sim_k$;

5 ponderar essas distâncias pela matriz de covariâncias da respectiva nuvem de pontos $non.dom.sim_k$;

6 escolher o k com a maior distância ponderada observada;

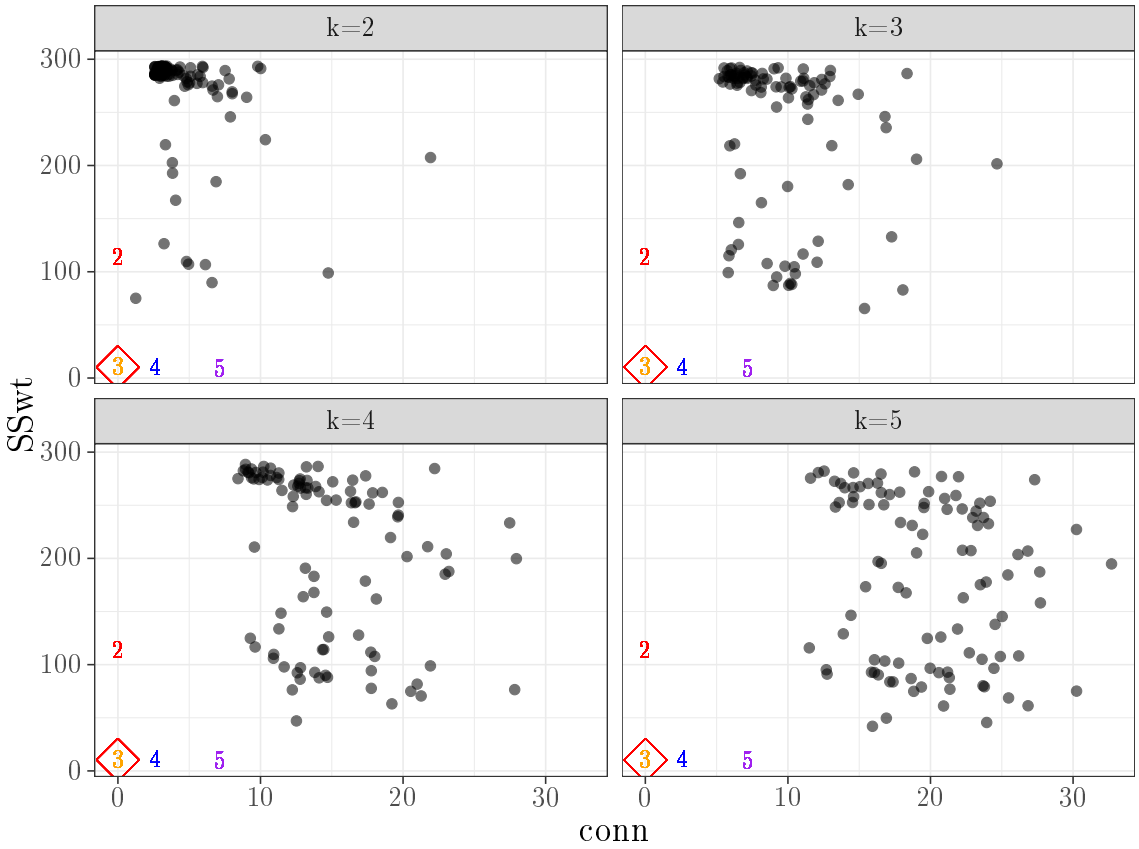

Figura 18 – Exemplo de determinação do k

Na figura 18, cada número plotado indica um ponto das estatísticas de teste dos dados reais. Em cada quadro, a nuvem de pontos pretos representa os pontos não-dominados das estatísticas de teste calculadas a partir das simulações para um k específico. O losango aponta o k escolhido ($k = 3$) ao término do algoritmo. Note que a escolha de $k = 3$ deu-se justamente por este ponto ser o mais distante do centróide de sua respectiva nuvem, ponderando-se esta distância pela respectiva variância dos pontos. Note também que o ponto 2 é um ponto dominado para as estatísticas de teste dos dados reais. Por isso, durante a execução do algoritmo, acaba nem sendo considerado como uma solução viável no caso ilustrado.

4 RESULTADOS

Tendo em vista que a implementação do algoritmo proposto é relativamente complexa, devido aos outros algoritmos e metodologias envolvidas, dispendeu-se um tempo considerável, mas necessário, na implementação do método e em ajustes tanto de ordem computacional quanto conceitual. Isso impediu que o número de testes realizado até a escrita desse trabalho fosse, a nosso ver, suficiente.

Para a implementação, inicialmente, tentou-se uma abordagem com *kmeans* e uma função uniobjetivo baseada apenas em soma de quadrados sem que houvesse êxito na identificação do número de clusters. Após isso, optou-se pela implementado em código **C++** do algoritmo de Prim para *MST*. Mesmo sendo uma linguagem rápida, os testes iniciais não permitiam utilizar dados acima de algumas centenas de observações, devido ao código pouco eficiente que escrevemos. Só então, passou-se a utilizar o algoritmo de *Minimum Spanning Trees Dual Tree Boruvka* da biblioteca **mlpack** de **C++**. Em todas essas abordagens, a ligação dos códigos em **C++** e **R** via API demandou uma grande quantidade de tempo.

Não obstante os contratempos de ordem computacional, os poucos testes realizados até o momento indicam que o algoritmo funciona de forma satisfatória. Utilizamos dados elípticos gerados manualmente e automaticamente a partir de distribuições normais e dados não convencionais gerados manualmente, como espirais, círculos e misturas de figuras. Procurou-se variar o número de *clusters* ($k = 2, 3, 4, 10$), os tamanhos de cada grupo (iguais e desiguais), bem como a dimensionalidade dos dados, tendo sido testados dados em $2, 3$ e 10 dimensões.

4.1 Desempenho geral

Na Tabela 1 indicamos os testes realizados e os resultados obtidos. As linhas em cinza indicam os acertos e as brancas os erros com relação ao número real de *clusters* e o k retornado pelo algoritmo proposto. Nas colunas, temos as seguintes informações:

- `type`: tipos de *clusters* presentes nos dados;
- `gen`: método utilizado para gerar os dados;
- `seed`: semente utilizada na confecção;

- `dim`: dimensão ou número de variáveis nos dados;
- `N`: número de observações nos dados;
- `size`: indicação se os tamanhos dos *clusters* são iguais ou não;
- `kmax`: parâmetro indicando o número máximo de *clusters* a ser testado no algoritmo;
- `n_neig`: parâmetro indicando o número de vizinhos mais próximos a se considerar para a identificação de links interessantes;
- `n_sim`: número de hipercubos ou simulações a serem geradas;
- `clust`: número de *clusters* originalmente presente nos dados;
- `result`: k retornado após execução do algoritmo;

Tabela 1 – Resultados dos testes

#	type	gen	seed	dim	N	size	$kmax$	n_neig	n_sim	clust	result
1	elipse	manual	1984	2d	66	$=$	5	8	100	3	$k = 3$
2	elipse	auto	12	2d	236	$\neq$	5	8	400	2	$k = 2$
3	elipse	manual	1984	2d	30	$=$	5	3	400	2	$k = 2$
4	elipse	manual	1984	2d	100	$=$	5	8	100	2	$k = 2$
5a	elipse	manual	1984	2d	1050	$=$	5	8	100	2	$k = 2$
5b	elipse	manual	1984	2d	1050	$=$	5	3	400	2	$k = 2$
6	elipse	*manual	1984	2d	620	$\neq$	6	10	1000	3	fail.MST
7	elipse	manual	123	2d	750	$=$	5	25	100	3	$k = 3$
8	elipse	auto	1984	3d	315	$\neq$	5	8	400	3	$k = 3$
9	elipse	auto	123	3d	520	$\neq$	6	7	400	4	fail.MST
10	elipse	auto	124	3d	633	$\neq$	5	7	400	4	$k = 4$
11a	elipse	auto	12	3d	383	$\neq$	6	5	400	4	$k \geq 6$
11b	elipse	auto	12	3d	383	$\neq$	6	5	1000	4	$k \geq 6$
11c	elipse	auto	12	3d	383	$\neq$	6	7	1000	4	$k = 4$
12a	elipse	**auto	-	10d	838	$\neq$	5	3	400	4	$k \geq 5$
12b	elipse	**auto	-	10d	838	$\neq$	6	7	1000	4	fail.MST
13a	anelar + elip.	*manual	1984	2d	550	$=$	5	7	1000	2	$k = 4$
13b	anelar + elip.	*manual	1984	2d	550	$=$	5	25	1000	2	$k = 4$
14a	anelar + elip.	*manual	1984	2d	1000	$=$	5	25	400	4	$k >= 5$
14b	anelar + elip.	*manual	1984	2d	1000	$=$	6	10	1000	4	$k = 5$
15a	***espiral	manual	-	2d	1000	$=$	5	7	400	3	$k = 4$
15b	***espiral	manual	-	2d	300	$=$	5	20	300	3	$k = 2$
15c	***espiral	manual	-	2d	300	$=$	5	15	300	3	$k = 3$
15d	***espiral	manual	-	2d	300	$=$	5	15	1000	3	$k = 3$
16	***espiral	manual	1984	2d	300	$=$	5	20	300	3	$k = 2$
17	alongado	manual	123	2d	300	$=$	5	30	300	4	$k = 4$
18	along. + elip	manual	123	2d	750	$=$	5	25	100	3	$k = 3$

**dados de David Robinson(2015) - http://varianceexplained.org/r/kmeans-free-lunch/*
***dados de Handl e Knowles (2007)*
****espirais geradas com o pacote KODAMA para linguagem* ***R***

Pelos testes realizados, o melhor desempenho do algoritmo ocorre quando os dados apresentam *clusters* em formato elipsóide **sem outliers significativos**. Em dados com presença de *outliers*, a dificuldade ocorre ainda durante o *MST based clustering*, uma vez que o método de corte via links interessantes acaba identificando *outliers* como pequenos *clusters* separados. Daí a identificação de *fail.MST*. Como o método de clusterização falha mesmo que informemos o k verdadeiro, não convém rodar completamente o algoritmo de determinação do número de grupos. Mesmo que eventualmente se acertasse o k, tratar-se-ia de uma mera casualidade.

Nota-se também o mau desempenho do algoritmo no caso de *clusters* anelares

ou semi-anelares (em 2 dimensões), principalmente quando há grupo(s) que tendem a ocupar as margens da figura. Suspeita-se que o desempenho ruim nestes casos esteja ligado ao fato de *clusters* anelares ficarem mais próximos dos hipercubos que são simulados, em termos de preenchimento de área. Desse modo, a diferença entre as estatísticas de teste dos dados originais e das simulações serão maiores quando temos soluções de agrupamentos nos dados que quebram *clusters anelares*, o que não corresponde à realidade. Cabe, no entanto, mesmo nestes casos, uma menção ao desempenho de nosso algoritmo de clusterização, o qual se mostra capaz de encontrar corretamente os agrupamentos quando informamos o k verdadeiro. Na maioria dos casos em que o algoritmo proposto para determinação do número de *clusters* erra, nosso algoritmo de *MST based clustering* identifica os agrupamentos originais perfeitamente ao passarmos o valor correto de k.

Alguns testes que inicialmente resultaram em erro foram repetidos com outros parâmetros, tendo apresentado a solução esperada posteriormente. Um outro aspecto positivo do algoritmo é que, mesmo nos casos em que erra, a convergência para determinado k ocorre rapidamente, com cerca de 300, 400 ou até mesmo 100 simulações apenas. Abaixo, apresentamos as figuras comparando o resultado esperado e a solução apresentada pelo algoritmo proposto.

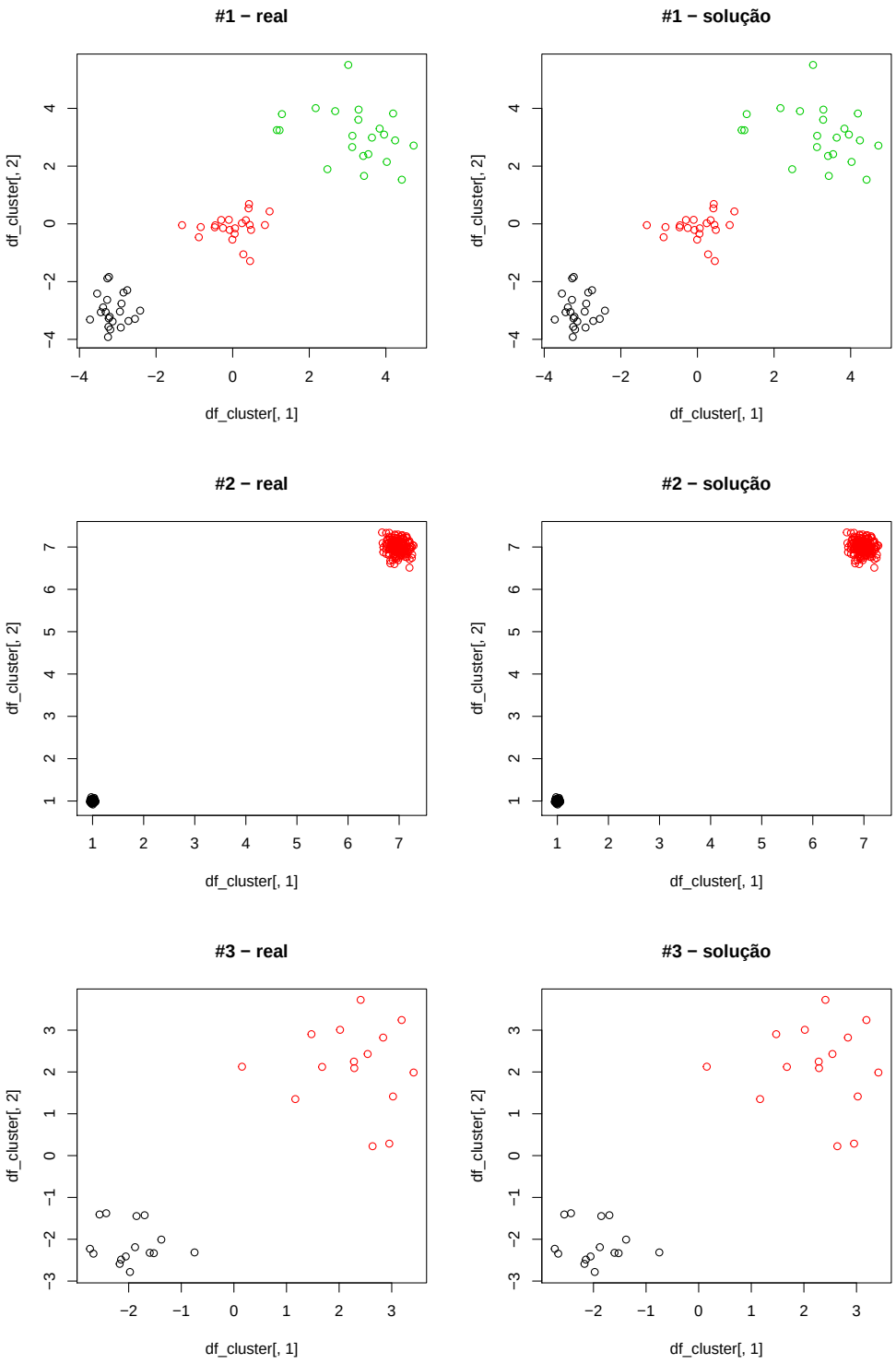

Figura 19 – Testes #1, #2 e #3

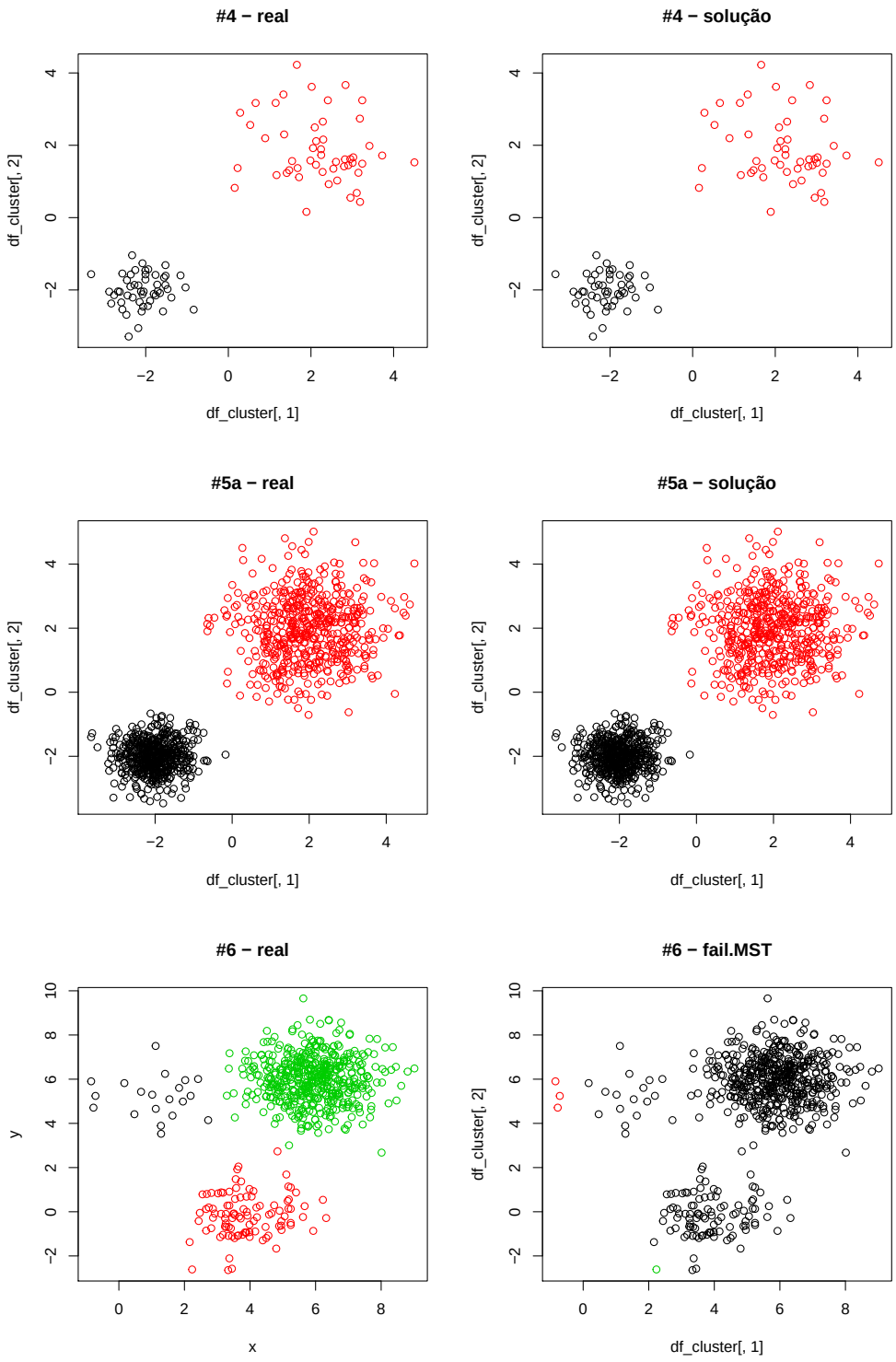

Figura 20 – Testes #4, #5a e #6

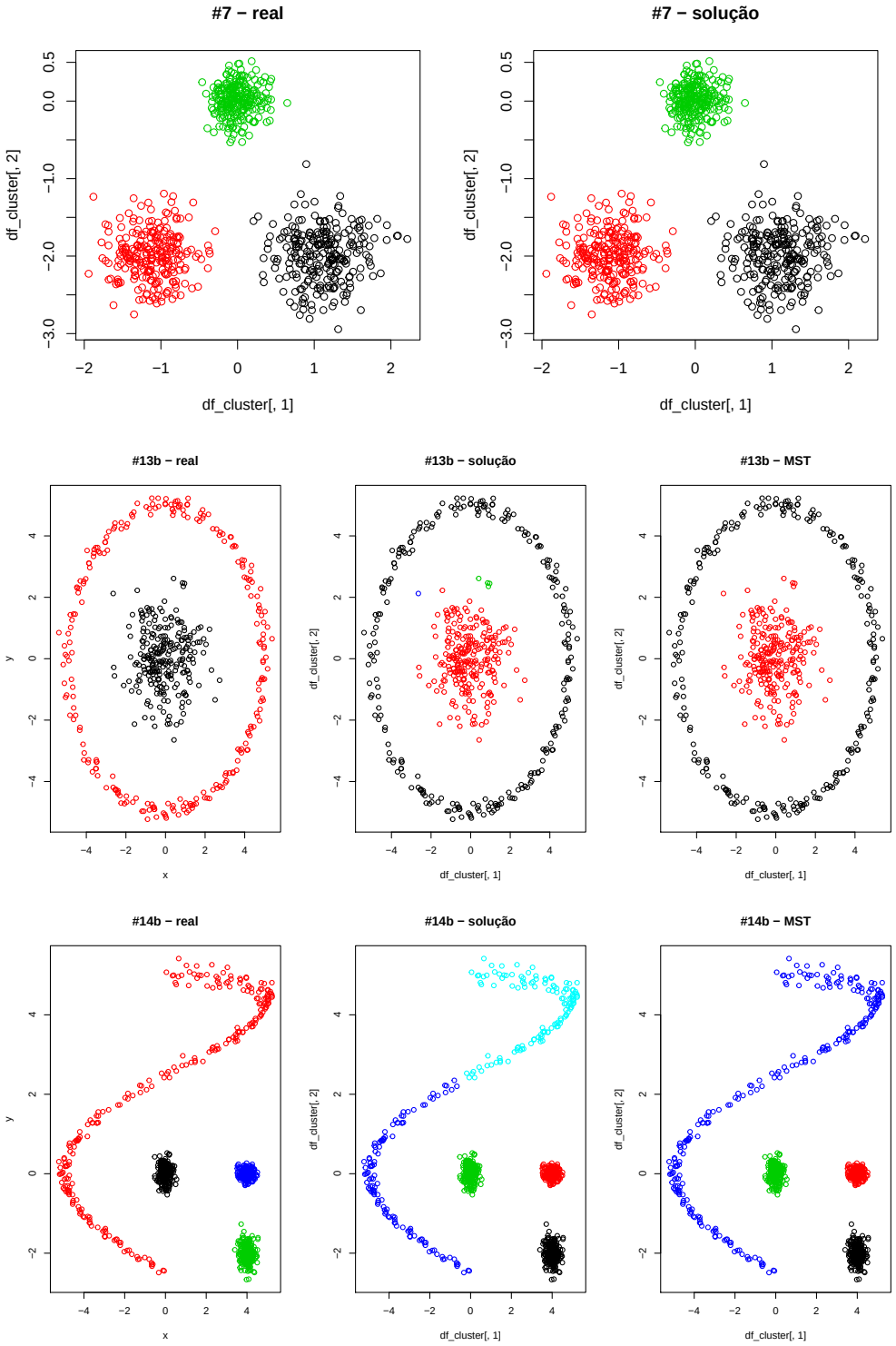

Figura 21 – Testes #7, #13b e #14b

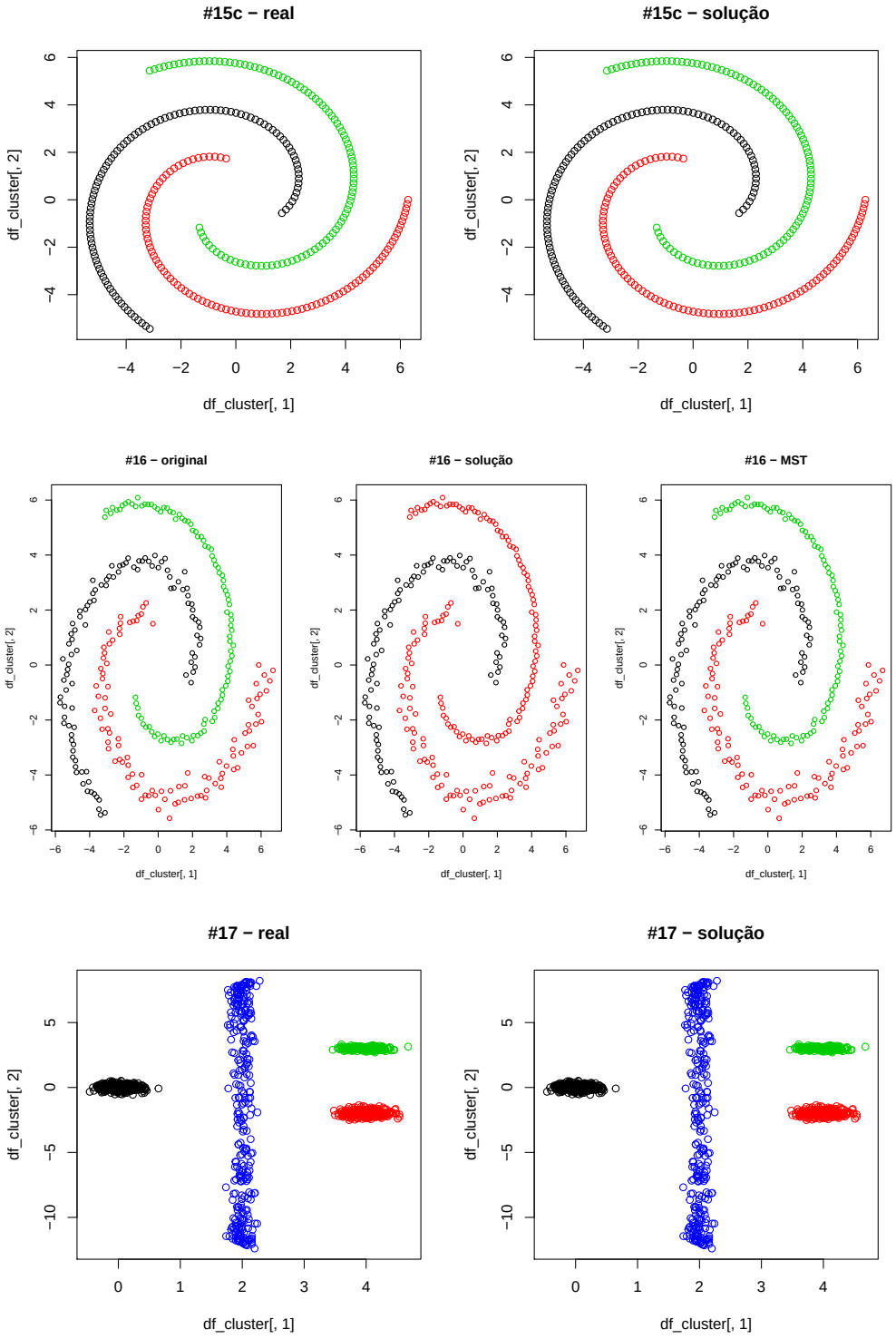

Figura 22 – Testes #15c, #16 e #17

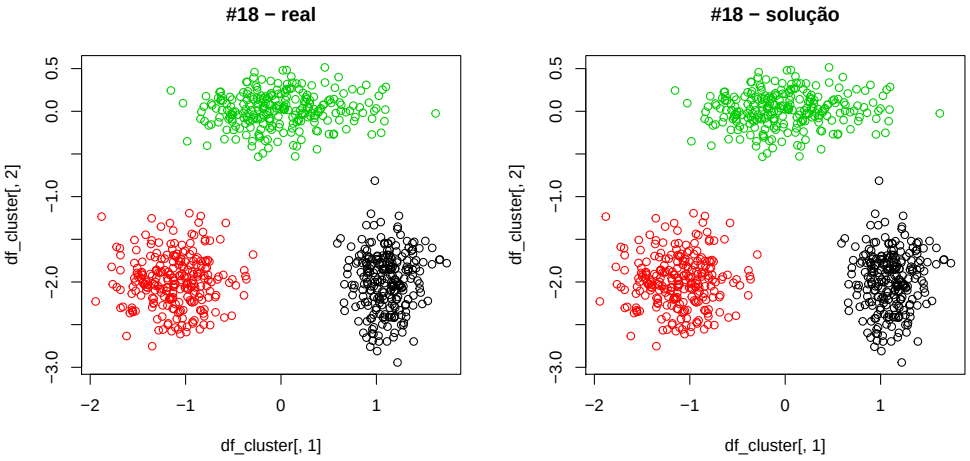

Figura 23 – Teste #18

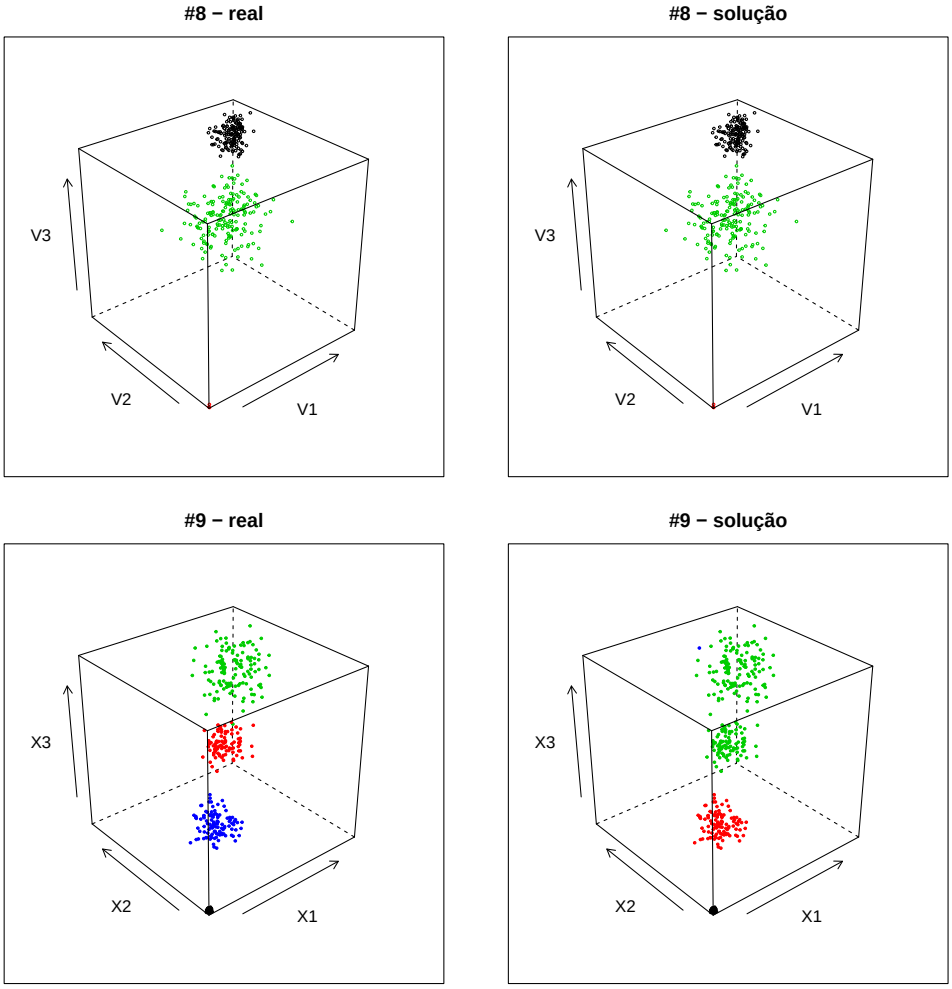

Figura 24 – Testes #8 e #9

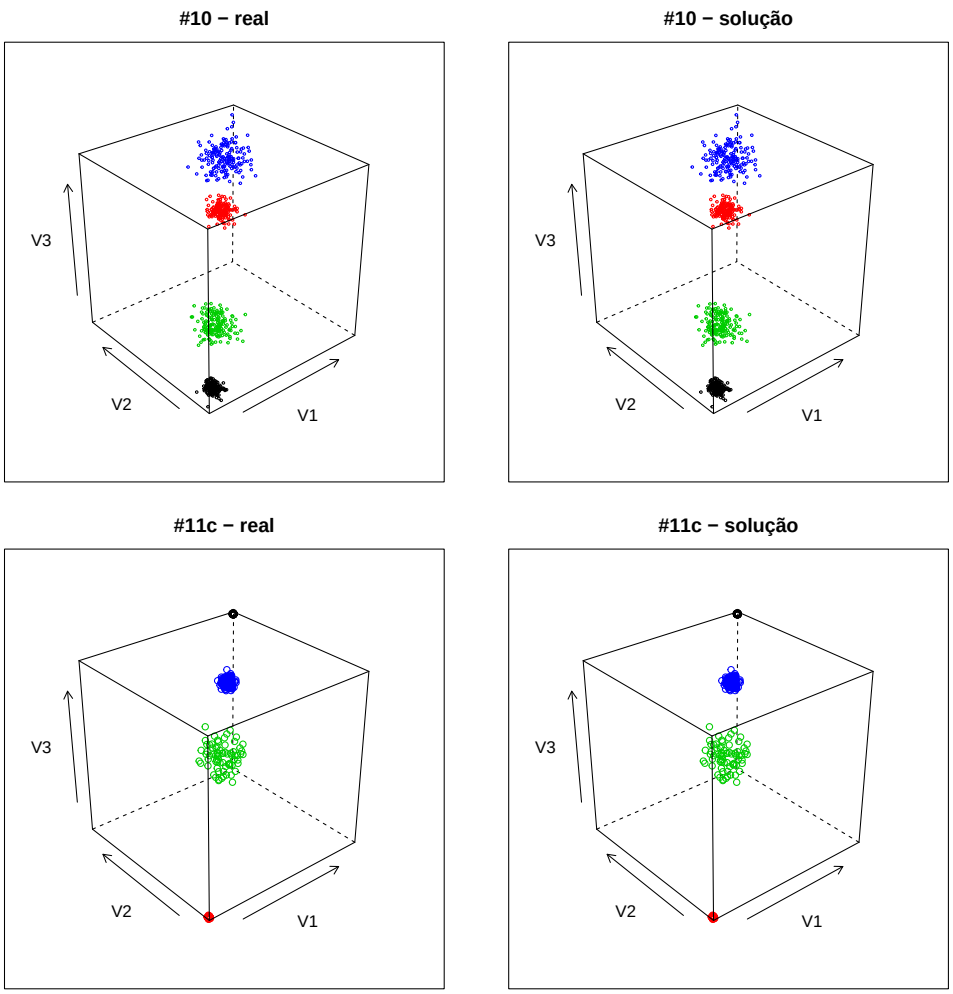

Figura 25 – Teste #10 e #11c

4.2 Restrições

Em termos computacionais, o algoritmo fica relativamente lento conforme aumentamos o número de simulações, o k máximo a ser testado ou mesmo o número máximo de vizinhos mais próximos a ser considerado. Dados relativamente grandes exigirão um maior número de vizinhos mais próximos e aumentará o tempo de processamento. Da mesma forma, dados em maiores dimensões exigirão maior número de simulações contribuindo para tornar o algoritmo mais lento. Isso se deve a uma implementação pouco eficiente em termos computacionais de uma função específica que confeccionamos para rotular os dados após o corte dos links interessantes.

Reconhecemos também que a necessidade de se especificar um número de vizi-

nhos mais próximos acaba sendo uma desvantagem do algoritmo. Dependendo do equílibrio entre este valor e o tamanho dos dados, a solução gerada, como vimos na Tabela 1, pode ser diferente.

5 CONCLUSÃO

O algoritmo para determinação do número k de *clusters* proposto neste trabalho apresentou resultado satisfatório nos casos envolvendo agrupamentos elipsóides (em 2 e 3 dimensões) e alongados. Os casos de agrupamentos anelares, em maiores dimensões ou na presença de outliers dificultaram seu desempenho.

Propõe-se que estudos futuros avaliem a viabilidade de se utilizar o conceito de hipercubo como hipótese nula, bem como a possibidade de acrescentar ou testar outras estatísticas de teste na função objetivo além das que foram utilizadas neste estudo. São necessários também mais testes a fim de avaliar o desempenho do algoritmo em outras configurações de agrupamento.

Não obstante os problemas encontrados, ressalta-se o excelente desempenho de nosso algoritmo de *MST based clustering* para diversos tipos de agrupamentos, o que o torna uma alternativa viável em relação a alguns algoritmos de clusterização como *kmeans*, *single linkage* e o próprio *GMM (Gaussian Mixture Models)* em alguns casos específicos. Pretende-se realizar alguns ajustes neste algoritmo em particular, como otimização em termos computacionais e em relação à identificação de outliers nos dados para que então o disponibilizemos por meio de um pacote para a linguagem **R** no **CRAN**.

REFERÊNCIAS

AUFFERMANN, W. F.; NGAN, S. C.; HU, X. Cluster significance testing using the bootstrap. *NeuroImage*, v. 17, p. 583–591, 2002.

BORRIES, G. F. von. *Slides da Disciplina Análise Multivariada I*. 2017. Curso de Graduação em Estatística - UnB.

BOTTOMLEY, P.; NAIRN, A. Blinded by science: The managerial consequences of inadequately validated cluster analysis solutions. *International Journal of Market Research*, v. 46, n. 2, 2004.

CHARRAD, M. et al. Nbclust: An R package for determining the relevant number of clusters in a data set. *Journal of Statistical Software*, v. 61, n. 6, October 2014.

CHARTRAND, G. *Introductory Graph Theory*. [S.l.]: Dover Publications, Inc. New York, 1975.

CURTIN, R. R. et al. MLPACK: A scalable C++ machine learning library. *Journal of Machine Learning Research*, v. 14, 2013.

EDDELBUETTEL, D.; FRANÇOIS, R. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, v. 40, n. 8, April 2011.

EVERITT, B. S. et al. *Cluster Analysis*. 5th edition. ed. [S.l.: s.n.], 2011.

FUENTES, C.; CASELLA, G. Testing for the existence of clusters. p. 115–157, Jul 2009. Sort (Barc). 33(2).

GAN, G.; MA, C.; WU, J. *Data Clustering: Theory, Algorithm and Applications*. [S.l.: s.n.], 2007.

GRAHAM, R. L.; HELL, P. On the history of the minimum spanning tree problem. *Ann. Hist. Comput.*, p. 7:1, 43–57, 1985.

HALKIDI, M.; BATISTAKIS, Y.; VAZIRGIANNIS, M. On clustering validation techniques. *Journal of Intelligent Information Systems*, v. 17, n. 2/3, p. 107–145, 2001.

HANDL, J.; KNOWLES, J. An evolutionary approach to objective clustering. *IEEE Transactions on Evolutionary Computation*, v. 11, n. 1, Feb 2007.

JAYAWANT, P.; GLAVIN, K. Minimum spanning trees. *INVOLVE*, v. 2, n. 4, p. 439–450, 2009.

LIU, Y. et al. Statistical significance of clustering for high-dimension, low-sample size data. *Journal of the American Statistical Association*, v. 103, n. 483, p. 1281–1293, Sep 2008.

MARCH, W. B.; RAM, P.; GRAY, A. G. Fast euclidian minimum spanning tree: algorithm analysis, and applications. *16th ACM SIGKDD international conference on Knowledge discovery and data mining*, July 25-28 2010. Washington, DC, USA.

MILIGAN, G. W.; COOPER, M. C. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, v. 50, n. 2, p. 159–179, June 1985.

MISHRA, K. K.; HARIT, S. A tree algorithm for finding the non dominated set in multi objective optimization. *International Journal of Computer Applications (0975-8887)*, v. 1, n. 25, p. 35–39, 2010.

MLPACK website. <https://www.mlpack.org/>. Acessado em jun 2018.

OKSANEN, J. *Cluster Analysis: Tutorial with R*. [S.l.], 2014. Disponivel em http://cc.oulu.fi/ jarioksa/opetus/metodi/sessio3.pdf.

POP, P. C. *The Generalized Minimum Spanning Tree Problem*. Tese (Doutorado) — University of Twente, 2002.

ROBINSON, D. *K-means clustering is not a free lunch*. 2015. <http://varianceexplained.org/r/kmeans-free-lunch/>. Acessado em jun 2018.

SARLE, W. S. *Cubic Clustering Criterion*. [S.l.], 1983. v. 46, n. A-108. Cary, NC.: SAS Institute, 1983, 56 pp.

WICKHAM, H. *Advanced R*. [S.l.]: Chapman & Hall/CRC Press, 2014.

WOLPERT, D. G.; MACREADY, W. G. Coevolutionary free lunches. *IEEE Transactions on Evolutionary Computation*, v. 9, n. 6, p. 721–735, January 2006.