UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS DE DOIS VIZINHOS
CURSO DE ESPECIALIZAÇÃO EM CIÊNCIA DE DADOS

FELIPE PEDROSO DE LIMA BRUSSE

INCLUINDO UMA POLÍTICA DE CONSTRUÇÃO DE OBJETOS ESPACIAIS FUZZY NO PACOTE FSR

TRABALHO DE CONCLUSÃO DE CURSO DE ESPECIALIZAÇÃO

DOIS VIZINHOS
2021

FELIPE PEDROSO DE LIMA BRUSSE

INCLUINDO UMA POLÍTICA DE CONSTRUÇÃO DE OBJETOS ESPACIAIS FUZZY NO PACOTE FSR

Trabalho de Conclusão de Curso de Especialização apresentado ao Curso de Especialização em Ciência de Dados da Universidade Tecnológica Federal do Paraná, como requisito para a obtenção do título de Especialista em Ciência de Dados.

Orientador: Prof. Dr. Anderson Chaves Carniel

DOIS VIZINHOS
2021

FELIPE PEDROSO DE LIMA BRUSSE

INCLUINDO UMA POLÍTICA DE CONSTRUÇÃO DE OBJETOS ESPACIAIS FUZZY NO PACOTE FSR

Trabalho de Conclusão de Curso de Especialização apresentado ao Curso de Especialização em Ciência de Dados da Universidade Tecnológica Federal do Paraná, como requisito para a obtenção do título de Especialista em Ciência de Dados.

Data de Aprovação: 30/dezembro/2021

Anderson Chaves Carniel
Doutorado
Universidade Federal de São Carlos

Yuri Kaszubowski Lopes
Doutorado
Universidade do Estado de Santa Catarina

Rafael Alves Paes de Oliveira
Doutorado
Universidade Tecnológica Federal do Paraná - Câmpus Dois Vizinhos

DOIS VIZINHOS
2021

Dedico esse Trabalho de Conclusão de Curso à minha esposa, que concedeu todo o apoio para realizá-lo, teve paciência ao abrir mão dos momentos em família e carinho nos momentos que precisei de força para realizá-lo.

Dedico também esse TCC aos meus pais e ao meu irmão, que sempre me apoiaram e confiaram nos meus estudos.

Uma dedicação especial também ao prof. Dr. Anderson Chaves Carniel, que teve muita paciência e dedicação, ao me ensinar muito sobre R, banco de dados espaciais e Álgebra Espacial *Plateau*.

RESUMO

Atualmente, os geocientistas têm mostrado interesse em modelar fenômenos espaciais com tipos de dados imprecisos, assim, têm-se utilizado a teoria de conjunto *fuzzy* como um arcabouço para modelar esses tipos de dados imprecisos, através de objetos espaciais *fuzzy*. Nesse contexto, existem modelos para definir formalmente objetos espaciais *fuzzy* bem como álgebras formais, como a Álgebra Espacial *Plateau*, que implementam os conceitos formais dos objetos espaciais *fuzzy*. Para construir tais objetos, há uma metodologia formada por duas fases, sendo a primeira fase com políticas de *fuzzificação* e a segunda fase com políticas de construção baseadas em diagramas de *Voronoi*, triangulação *Delaunay* e *convex hull*. O pacote *fsr*, para a linguagem de programação R, implementa os tipos de dados espaciais *fuzzy* baseados na Álgebra Espacial *Plateau* bem como suas operações. Na arquitetura do pacote *fsr*, implementou-se duas políticas de construção, a baseada no diagrama de *Voronoi* e na triangulação *Delaunay*, e até o momento, não foi desenvolvida a política de construção *convex hull*. Nesse sentido, o objetivo desse Trabalho de Conclusão de Curso é implementar a política de construção baseado no *convex hull* no pacote *fsr*, e apresentar um estudo de caso para validar a implementação realizada. Como resultado, este trabalho apresenta o algoritmo de implementação da política *convex hull*, descrevendo seu funcionamento e acoplação no pacote *fsr*. Para a validação de seu funcionamento, simulou-se uma base de dados espaciais e quatro classes do fenômeno *fuzzy* temperatura. Este trabalho também apresenta um estudo de caso que contempla as três políticas de contrução para analisá-las. Concluí-se que o algoritmo desenvolvido atendeu os conceitos da Álgebra Espacial *Plateau* e aplicou as regras da política de construção *Convex hull*.

Palavras-chave: Banco de Dados Espaciais, Álgebra Espacial *Plateau*; Objetos Espaciais *Fuzzy*; *Convex hull*.

ABSTRACT

Currently, geoscientists have shown interest in modeling spatial phenomena with types of imprecise data, thus, fuzzy set theory has been used as a framework to model these types of imprecise data, through fuzzy spatial objects. In this context, there are models to formally define fuzzy spatial objects, as well as formal algebras, such as the Spatial Algebra Plateau, that implement the formal concepts of fuzzy spatial objects. To build such objects, there is a methodology composed of two phases, where the first phase with fuzzification policies, and the second phase with construction policies based on Voronoi diagrams, Delaunay triangulations, and convex hulls. The *fsr* package, for the R programming language, implements the fuzzy saptial data types based on the Spatial Plateau Algebra. In the *fsr* package architecture, two construction policies were implemented, the Voronoi Diagram policy and the Delaunay Triangulation policy, and so far, the Convex Hull policy has not been developed. In this sense, the objective of this work is to implement the construction policy based on the convex hull build in the *fsr* package, and present a case study to validate the implementation performed. As a result, this work presents the implemented algorithm for the convex hull policy, describing how it works and how it was coupled in the *fsr* package. To validate the implementation, we simulated a spatial database and four classes of the fuzzy temperature phenomenon. This work also presents a case study that includes the three construction policies, Voronoi Diagram, Dalaunay Triangulation, and Convex hul, to analyze them. We conclude that the developed algorithm met the concepts of Spatial Plateau Algebra and applied the convex hull construction policy rules.

Keywords: Spatial Databases, Spatial Plateau Algebra, Fuzzy Spatial Objects, Convex Hull.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

SIG	Sistemas de Informações Geográficas
AEP	Álgebra Espacial *Plateau*
TDE	Tipos de Dados Espaciais
TCC	Trabalho de Conclusão de Curso
0-D	Zero Dimensão
1-D	Uma Dimensão
2-D	Duas Dimensões
3-D	Três Dimensões

SUMÁRIO

1 INTRODUÇÃO

Os Sistemas de Informação Geográfica (SIG) e os banco de dados espaciais são ferramentas sofisticadas para representar ou gerenciar objetos espaciais *crisp*, que são caracterizados por terem localização exata e precisão na sua definição (CARNIEL; SCHNEIDER, 2019). Os sistemas de bancos de dados espaciais oferecem tipos de dados espaciais (TDE) em seu modelo de dados e linguagem para consulta (GÜTING, 1994) e esses tipos de dados espaciais são estruturas que modelam entidades geométricas no espaço, como pontos, linhas e regiões. Com isso, os tipos de dados espaciais para representar os objetos espaciais *crisp* correspondem a pontos *crisp*, linhas *crisp* e regiões *crisp*, e são implementados junto com as suas operações geométricas como união, intersecção e diferença.

Entretanto, atualmente os geocientistas têm mostrado interesse em modelar fenômenos espaciais com tipos de dados imprecisos, como por exemplo áreas poluídas ou zonas de temperaturas. Esses tipos de dados espaciais vêm sendo representados pela lógica *fuzzy* e são caracterizados pela inexatidão de sua localidade ou vagueza nas bordas, e não podem ser representados por objetos espaciais *crisp* (CARNIEL; SCHNEIDER, 2018).

A teoria de conjuntos *fuzzy* foi introduzida em 1965 (ZADEH, 1965) pelo matemático Lotfi Asker Zadeh, e teve o objetivo em dar tratamento matemático a certos termos linguísticos como "aproximadamente" ou "em torno de", e vem sendo usada como ferramenta para formulação de modelos em vários campos da ciência (ZIMMERMANN, 2010). Um conjunto *fuzzy* é uma classe de objetos com graus de pertinência, e esse conjunto é caracterizado por uma função que atribui a cada objeto um grau de pertinência, variando entre zero e um. Na teoria de conjuntos clássica, em um conjunto de elementos (universo), cada elemento de um subconjunto possui uma pertinência definida por pertencer ou não ao conjunto (universo), sendo essa decisão *crisp*, rígida e binária. Já na teoria de conjuntos *fuzzy*, a pertinência de cada elemento do subconjunto é relaxada, sendo que para cada elemento determina-se um grau de pertinência para o conjunto (universo). Com isso, nessa teoria existe uma pertinência parcial e um mesmo elemento pode ter diferentes graus de pertinência em subconjuntos diferentes (CHEN; RAZAK; GARIBALDI, 2020).

Nesse sentido, os geocientistas e os SIG têm utilizado a teoria de conjuntos *fuzzy* como um *framework* para modelar fenômenos espaciais imprecisos através de objetos espaciais *fuzzy*, como pontos *fuzzy*, linhas *fuzzy* e regiões *fuzzy*, junto com as suas operações espaciais *fuzzy* (DILO; de By; STEIN, 2007; CARNIEL; SCHNEIDER, 2018). Nesse contexto, um modelo para definir formalmente as relações topológicas *fuzzy* entre objetos espaciais *fuzzy* foi proposto por Carniel e Schneider (2016), e em seguida, dois problemas surgiram e foram objetos de estudo no trabalho de Carniel e Schneider (2018), sendo o primeiro referente a como implementar os conceitos formais dos objetos espaciais *fuzzy* e o segundo referente a como realizar consultas nos bancos de dados espaciais. Para isso, os autores propuseram como solução a Álgebra

Espacial *Plateau* (AEP), como uma álgebra executável, ao aproximar os conceitos dos pontos *fuzzy*, linhas *fuzzy* e regiões *fuzzy*, e demonstraram como a AEP pode ser implementada nos esquemas dos bancos de dados espaciais.

Em progresso, Carniel e Schneider (2019) desenvolveram uma metodologia para criar objetos de regiões *fuzzy* com conjuntos de dados espaciais reais. Nessa metodologia de extração de dados, os autores propuseram duas etapas para a extração dos dados, sendo a primeira etapa a política de *fuzzificação* e a segunda etapa a política de construção. A etapa de *fuzzificação* consiste em aplicar os conceitos da teoria de conjutos *fuzzy* e atribuir graus de pertinência para cada ponto da base de dados espacial. Para isso, os autores propuseram três políticas de *fuzzificação*, a política de conjuntos *fuzzy*, a política de clusterização *fuzzy* e a política de classificação *fuzzy*.

Já na etapa de construção, utiliza-se os resultados da etapa de *fuzzificação* e com a aplicação de algoritmos espaciais agrupa-se os pontos com características semelhantes para a formação da região *plateau* do objeto. Os autores propuseram três algoritmos espaciais com políticas distintas de construção: a política do Diagrama de *Voronoi*, a política da Triangulação *Delaunay*, e a política de *Convex hull*.

Posteriormente, devida a escassez de tecnologias que conectassem os SIG, sistemas de banco de dados espaciais, e os dados espaciais *fuzzy*, os autores e colaboradores do projeto desenvolveram um pacote denominado *fsr* para a linguagem de programação R, com o objetivo de implementar os tipos de dados espaciais *fuzzy*, as operações e os conceitos da AEP (CARNIEL et al., 2021). Na arquitetura do pacote *fsr*, os autores apresentaram funções para quatro módulos, denominados como Módulo Básico, Módulo de Construção, Módulo de Manipulação de Dados Espaciais *Fuzzy*, e Módulo de Inferência Espacial *Fuzzy*. Assim, para o Módulo de Construção, desenvolveu-se a função *spa_creator*, com duas políticas de construção, a política do Diagrama de *Voronoi* e a política da Triangulação *Delaunay*, e até o momento, não foi desenvolvida a política de construção *Convex hull*.

Nesse sentido, o objetivo desse Trabalho de Conclusão de Curso (TCC) é implementar a política de construção *Convex hull* no pacote *fsr*, conforme descrito no trabalho de Carniel e Schneider (2019), e apresentar um estudo de caso para validar a implementação realizada e uma comparação com as outras políticas de construção. Como resultado implementou-se essa política na forma de uma função escrita na linguagem de programação R e a sua inclusão na função *spa_creator* do pacote *fsr*.

Esta monografia está organizada da seguinte forma: Capítulo 2 descreve a revisão da literatura, com conceitos da teoria de conjuntos *fuzzy*, banco de dados espaciais, AEP, o pacote *fsr*, e as suas políticas de construção. Capítulo 3 apresenta os resultados obtidos e a exposição do estudo de caso. O Capítulo 4 destaca a conclusões e trabalhos futuros.

2 REVISÃO DE LITERATURA

2.1 Teoria de Conjuntos Fuzzy

A teoria de conjuntos *fuzzy* teve as suas primeiras publicações na década de 60 (ZIMMERMANN, 2010), com os autores Lotfi Zadeh em 1965 e com Joseph Goguen em 1969. De acordo com os autores, a motivação da teoria de conjuntos *fuzzy* deu-se em trazer a imprecisão, que está contida na linguagem e decisões humana, para a teoria de conjuntos, sendo uma estrutura que fornece uma maneira natural de lidar com essa classe de fenômenos imprecisos.

O conceito de grau de pertinência foi a chave para o desenvolvimento da teoria de conjuntos *fuzzy* (ZADEH, 2015) e a expectativa é que essa teoria fosse bem recebida por áreas das comunidades das ciências sociais, entretanto, a teoria foi bem recebida pela comunidade de engenheiros, principalmente no Japão. Nas décadas de 60 e 70, a teoria de conjuntos *fuzzy* teve uma lenta aceitação, entretanto, na metade da década de 70, com as suas primeiras publicações práticas, aumentou-se consideravelmente o interesse nessa teoria (ZIMMERMANN, 2010). Nesse sentido, na década de 80, aplicações práticas bem-sucedidas, principalmente no Japão, como a utilização da teoria de conjuntos *fuzzy* nos sistemas do metrô, câmeras de vídeos, e máquinas de lavar por exemplo, fizeram crescer o número de publicações nessa área, passando de 4.000 publicações em 1984, para mais de 30.000 no ano de 2.000, e de maneira geral, a teoria de conjuntos *fuzzy* desenvolveu-se nesse período em duas linhas, como uma teoria formal ou como uma tecnologia difusa, orientada para a aplicação (ZIMMERMANN, 2010).

Conforme Zadeh (1965), um conjunto *fuzzy* é uma classe de objetos com graus de pertinência, e esse conjunto é caracterizado por uma função que atribui a cada objeto um grau de pertinência variando entre $[0,1]$. Assim, seja X um conjunto de pontos (objetos) e A um conjunto *fuzzy* de X, esse conjunto é caracterizado por uma função de pertinência $\mu_A(x)$, que associa para cada ponto de A, um grau de pertinência em um intervalo $[0,1]$. Sendo assim, quanto mais próximo de 1 for o grau de pertinência do ponto, maior a sua pertinência a X.

Chen, Razak e Garibaldi (2020) propõem um pacote para a linguagem R que implementa diversos tipos comuns de funções de pertinência, para determinar como será a distribuição dos graus de pertinência do conjunto *fuzzy*, como a função de pertinência do tipo *singleton*, triangular, trapezoidal e a *gaussiana*. A função de pertinência do tipo *singleton* retorna o valor 1 para um único elemento e o valor 0 para os elementos restantes, sendo esse tipo de função de pertinência equivalente a um conjunto *crisp*, pois há um único valor com grau de pertinência igual a 1.

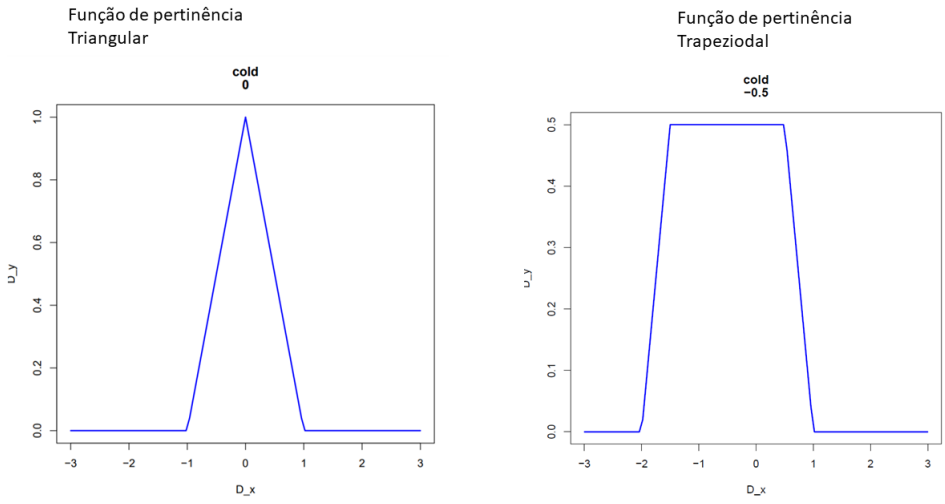
A função do tipo triangular tem a sua representação matemática como $A(x; a, b, c) = max(min((x-a)/(b-a), (c-x)/(c-b)), 0)$ e é baseada em três parâmetros (a, b e c).

Se os parâmetros forem de valores diferentes, o grau de pertinência cresce linearmente de a para b, e decresce linearmente de b para c, e caso a e b, ou b e c forem iguais, forma-se apenas uma função linear. A função trapezoidal tem a sua representação matemática como $A(x; a, b, c, d) = max(min((x - a)/(b - a), 1, (d - x)/(d - c)), 0)$ e é baseada em quatro parâmetros (a, b, c, d), e é definida por uma faixa de valores com os graus de pertinência igual a 1. Já a função *gaussiana* é representada por $A(x; a, b) = exp(-(x - b)^2/(2 * a^2))$ e possui dois parâmetros (a, b), onde *a* determina o desvio padrão e *b* determina a média.

A Figura 1 apresenta como exemplo a função de pertinência triangular e a função de pertinência trapezoidal. Na função de pertinência do tipo triangular, os parâmetros são $a = -1$, $b = 0$ e $c = 1$. Observa-se que no parâmetro *a*, o grau de pertinência é 0, ou seja, esse valor não pertence a esse conjunto, já para o parâmetro *b*, o grau de pertinência é 1, sendo que esse valor pertence totalmente ao conjunto. No parâmetro *c*, o grau de pertinência também é 0, sendo que esse valor não pertence ao conjunto. Observa-se também que o grau de pertinência cresce linearmente de *a* para *b*, e decresce linearmente de *b* para *c*.

Já no exemplo de função de pertinência do tipo trapezoidal, os parâmetros são $a = -2$, $b = -1{,}8$, $c = 0{,}8$, e $d = 1$. Observa-se que para os parâmetros *a* e *d*, o grau de pertinência é 0, e para os parâmetros *b* e *c*, existe uma faixa linear, em que o grau de pertinência é 0,5.

Figura 1 – Exemplos de Funções de Pertinência



Para as operações entre conjuntos *fuzzy*, existem três operações básicas, a união entre conjuntos *fuzzy*, a intersecção e o complemento entre os conjuntos *fuzzy*. A união entre dois conjuntos *fuzzy* é calculada através do valor máximo entre os graus de pertinência para cada um dos valores dos conjuntos. A intersecção entre dois conjuntos *fuzzy* é calculada entre o menor valor dos graus de pertinência para cada um dos valores dos conjuntos e o complemento entre dois conjuntos *fuzzy* é calculado pela diferença entre 1 e o grau de pertiência do conjunto *fuzzy*.

2.2 Banco de Dados Espaciais

Em diversos campos do conhecimento é necessário gerenciar dados relacionados ao espaço, e esse espaço de interesse, conforme descreve (GÜTING, 1994), pode ser uma parte da superfície terrestre, ou até um projeto 3-D de cadeias de moléculas de proteína. Nesse sentido, desde o surgimento dos sistemas de bancos de dados relacionais, tem-se tentado gerenciar dados espaciais nesses sistemas, e a partir de 1989, o termo sistema de banco de dados espacial tornou-se popular, devido aos simpósios realizados periodicamente.

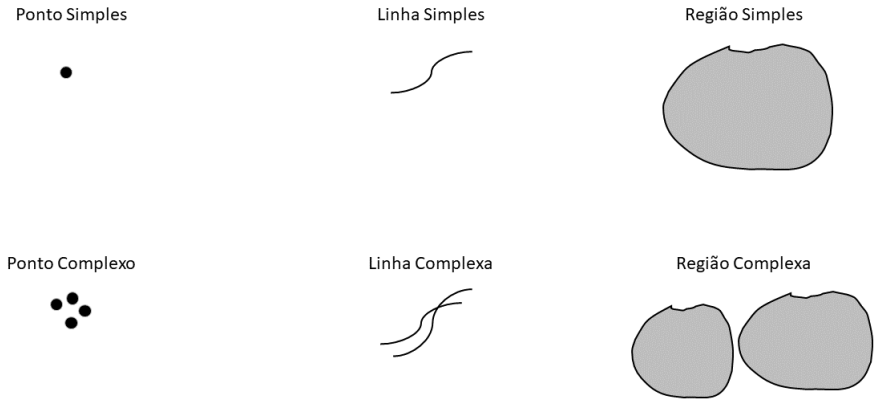
O trabalho de Güting (1994) disserta sobre três características dos sistemas de banco de dados espaciais. A primeira característica é que um sistema de banco de dados espaciais é um sistema de banco de dados com recursos adicionais para o processamento e gerenciamento de dados espaciais, assim, as informações espaciais sempre estarão conectadas com dados não espaciais. Na segunda característica, o autor descreve sobre os TDE, na qual modelam as entidades geométricas no espaço, como pontos, linhas e regiões, assim como as suas relações geométricas, como a união, interseção, áreas, distâncias, entre outras relações espaciais.

Já sobre a terceira característica, o autor descreve que um sistema de banco de dados espaciais deve ser capaz de recuperar grande coleções de objetos em algum espaço, dentro de uma área específica, assim, a indexação é obrigatória em um sistema de banco de dados espaciais. Além disso, o sistema de banco de dados espaciais deve ser capaz de realizar a conexão de objetos de diferentes classes por meio de algum relacionamento espacial.

Para a modelagem de dados espaciais, o espaço Euclidiano é frequentemente usado e Güting (1994) define dois conceitos importantes para a modelagem, objetos no espaço e espaço. Os objetos no espaço são entidades distintas disposta no espaço, com suas próprias descrições geométricas, como por exemplo cidades ou rios. Já o espaço é a descrição de cada ponto do espaço, para descrever o próprio espaço, como exemplo mapas temáticos de uso de solo.

Nesse sentido, vários modelos de dados e linguagens de consulta para dados espaciais foram propostos para formular e processar consultas espaciais e os tipos de dados espaciais, como ponto, linha ou região são os conceitos centrais desses modelos (SCHNEIDER; BEHR, 2006), pois fornecem abstrações para modelar estruturas de entidade geométricas, seus relacionamentos, propriedades e operações. Assim, esses tipos de dados espaciais podem assumir uma estrutura simples, com um único componente ou uma estrutura complexa, que contém uma quantidade finita de componentes, permitindo a representação da complexidade de fenômenos reais (CARNIEL, 2020). Um ponto representa um objeto espacial com apenas a sua localização no espaço, ou seja, um ponto é 0–D. Já uma linha é um objeto espacial com 1-D, pois não possui área e pode representar conexões no espaço, como por exemplo rios. Assim, uma região é uma abstração de um objeto que tem uma extensão no espaço com 2-D e geralmente representada por um polígono. Observa-se na Figura 2 os tipos de dados espaciais em suas estruturas simples e complexas.

Figura 2 – Exemplos de Tipos de Dados Espaciais Simples e Complexos



As relações espaciais descrevem como um objeto espacial está relacionado ou conectado com outro objeto espacial, e conforme (GÜTING, 1994; CARNIEL, 2020) existem diversas interpretações, como os relacionamentos topológicos, os relacionamentos métricos e as relações de direções. Utiliza-se frequentemente nas relações topológicas o modelo 9-interseções (SCHNEIDER; BEHR, 2006), na qual utiliza-se o conjunto de pontos e topologia para fornecer os pontos com combinações mutuamente exclusivas, das combinações de objetos espaciais simples ou complexos. O modelo de 9 interseções é baseado nas nove possíveis interseções de limite, interior e exterior, dos dois objetos espaciais, conforme os critérios topologicamente invariáveis de vazio ou não vazio, e como resultado, temos nove relacionamentos topológicos: *intersect*, *overlap*, *meet*, *disjoin*, *equal*, *inside*, *contains*, *covers*, e *coveredBy*.

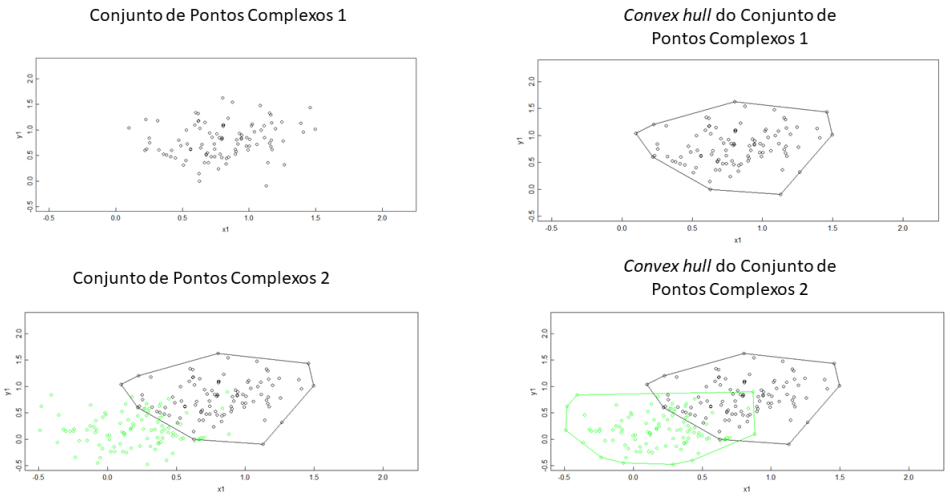
Além dos relacionamentos topológicos, encontra-se também as operações geométricas de conjunto, como a união, a interseção e a diferença. A união entre dois objetos espaciais A e B retorna todos os seus pontos. Já a intersecção entre dois objetos espaciais A e B retorna os seus pontos comuns. A operação de conjunto diferença entre dois objetos espaciais A e B retorna os pontos de A que não pertencem a B.

Outras operações espaciais são sugeridas na literatura (GÜTING, 1994), na quais manipulam conjuntos inteiros de objetos espaciais e com sua importância por manipular partições, como mapas temáticos. Entre algumas dessas operações, temos o *centroid*, que computa o centro de massa do objeto espacial consultado, a triangulação de *Delaunay*, que computa a triangulação dos pontos do objeto espacial, na qual os pontos fazem parte dos vértices do triangulo, o diagrama de *Voronoi*, que é uma região criada a partir de um ponto que abrange todos os pontos no plano mais próximos ao ponto original do que qualquer outro ponto do objeto espacial, e o *convex hull*.

O *convex hull* de um conjunto de pontos é a menor interseção que engloba todos os pontos desse conjunto. Assim, observa-se na Figura 3, dois conjuntos de pontos complexos, e

assim, forma-se os dois *convex hull* desses pontos, na qual cada *convex hull* forma um polígono contendo todos os pontos.

Figura 3 – Exemplo de formação de um *Convex hull*



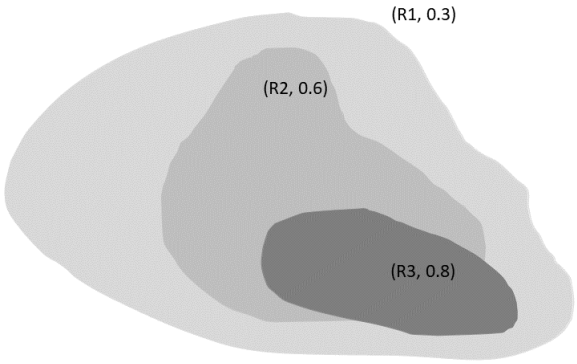
2.3 Álgebra Espacial Plateau

No trabalho de Carniel e Schneider (2018), a AEP foi proposta como uma solução para implementar os conceitos formais dos objetos espaciais *fuzzy* através dos tipos de dados espaciais *plateau*, que representam os pontos *fuzzy*, linhas *fuzzy* e regiões *fuzzy*, através dos pontos *plateau*, linhas *plateau* e regiões *plateau*.

Um objeto espacial *plateau* é formado por uma lista de pares, na qual cada par, denominado componente, contém um objeto espacial *crisp* e um grau de pertinência no intervalo de $]0, 1]$ (CARNIEL; SCHNEIDER, 2018), sendo que os objetos espaciais *crisp* dos componentes de um objeto espacial *plateau*, devem ser disjuntos ou adjacentes uns aos outros (CARNIEL; SCHNEIDER, 2018). O objeto espacial *plateau* contém em seu componente, uma sequência finita de pares, como por exemplo, as regiões adjacentes de uma classe (representação de um fenômeno), mas como o único elemento final, o grau de pertinência do objeto *plateau*, entretanto, o número de pares pode variar para diferentes objetos espaciais *plateau*, mas também pode ser zero, assim, formam-se os objetos espaciais *plateau* vazios.

A Figura 4 representa conceitualmente uma região *plateau* com três componentes com seus respectivos graus de pertinência.

Figura 4 – Exemplo de Região *Plateau*



No trabalho de Carniel e Schneider (2019), os autores propuseram uma metodologia genérica e flexível de extração de dados para criar objetos de regiões *fuzzy*, com dados espaciais reais. Essa metodologia é composta por duas fases, sendo a primeira, a fase de *fuzzificação*, com políticas para atribuir graus de pertinência para os dados espaciais, e a segunda fase é a política de construção, que utiliza o resultado da primeira fase, e com a aplicação de algoritmos espaciais, agrupa-se os pontos com características semelhantes para a formação da região *palteau* do objeto.

Na fase de *fuzzificação*, os autores propuseram três políticas distintas para atribuir os graus de pertinência aos conjuntos de dados espaciais. Genericamente, a política de *fuzzificação* tem como entrada um conjunto de dados espaciais, e retorna como saída, o mesmo conjunto de dados espaciais, entretanto, rotulados com graus de pertinência para cada classe que descreve o fenômeno, sendo essas classes definidas previamente pelo usuário (CARNIEL; SCHNEIDER, 2019).

A política de *fuzzificação* denominada política de conjunto *fuzzy* tem como primeira etapa o *design* de conjuntos *fuzzy*, com base nos valores numéricos anotados para cada ponto (CARNIEL; SCHNEIDER, 2019). Nesse sentido, cada conjunto *fuzzy* representa uma classe de um fenômeno descrito por um termo linguístico, como por exemplo, faixas de temperatura em um mapa térmico, como as classes quente e frio, e assim, para cada ponto do conjunto de dados espaciais de entrada, o seu grau de pertinência é avaliado para cada conjunto *fuzzy* criado, sendo assim, um ponto pode pertencer a mais de uma classe de fenômenos.

Já a política de *fuzzificação* denominada política de *clusterização fuzzy*, utiliza um algoritmo de agrupamento *fuzzy* para criar grupos de pontos com características semelhantes (CARNIEL; SCHNEIDER, 2019). Conforme os autores, essa política é útil quando o usuário não é capaz de definir previamente as classes do modelo de aplicação, e também destacam que é importante que o algoritmo de agrupamento seja baseado na teoria de conjuntos *fuzzy*, para

permitir que um mesmo ponto tenha diferentes graus de pertinência para as diferentes classes do modelo desenvolvido. Os autores propuseram também um parâmetro adicional *k*, para indicar o número de agrupamentos a ser definido pelo usuário, e o uso do clássico algoritmo de agrupamento *fuzzy* chamado *fuzzy c-means*.

Para a política de *fuzzificação* denominada política de classificação *fuzzy*, esta utiliza um algoritmo de classificação *fuzzy*, para atribuir graus de pertinência para cada ponto, considerando um conjunto de classes já conhecida. Assim como nas políticas de *fuzzificação* anteriormente descritas, na política de classificação *fuzzy*, um mesmo ponto pode ter graus de pertinência diferentes para cada classe de fenômeno para o modelo. Nesse sentido, os autores citam como exempo o clássico algoritmo de classificação para o modelo, a árvore de decisão *fuzzy*.

Para a fase de construção, os autores propuseram três políticas distintas que integram o resultado da etapa de *fuzzificação*, e agrupam os pontos com características semelhantes para formar a região *plateau* do objeto. A política de construção denominada política do diagrama de *Voronoi* é implementadas pelos seguintes passos. Para cada classe de fenômenos *fuzzy*, primeiro armazena-se todos os pontos com grau de pertinência maior do que 0 para a classe, e em seguida, o algoritmo cria um diagrama *Voronoi*, que consiste em um conjunto de células *Voronoi*. Cada célula *Voronoi* da classe é uma região criada a partir de um ponto armazenado, na qual abrange todos os pontos no plano mais próximos a esse ponto do que qualquer outro ponto do conjunto (CARNIEL; SCHNEIDER, 2019). Assim, para cada célula *Voronoi*, cria-se um par combinando a célula com a classe do fenômeno *fuzzy*, e cada par é anexado a região *plateau* do objeto.

Já a política de construção denominada política de triangulação *Delaunay* considera os seguintes passos. Para cada classe do fenômeno *fuzzy*, cria-se a triangulação *Delaunay* de todos os pontos com grau de pertinência maiores do que 0 da classe. Representa-se cada triangulo formado como uma matriz de três pares, na qual cada par armazena um ponto final do triângulo, junto com o seu grau de pertinência (CARNIEL; SCHNEIDER, 2019). Na sequência, percorre-se todos os triângulos formados das classes, e para cada triângulo forma-se um par, na qual cria-se uma variável m, com o cálculo da *t-norma* dos graus de pertinência dos pontos finais dos triângulos, e assim cria-se um triângulo com grau de pertinência único, e como resultado, adiciona-se o par em uma região *plateau*.

Para a política de construção denominada política *Convex hull*, aplica-se dois parâmetros adicionais de entrada, definidos pelo usuário. O primeiro parâmetro é um limite para agrupar os pontos com graus de pertinência próximos para uma determinada classe, sendo esse limite um valor numérico no intervalo $[0,1]$. O segundo parâmetro é um conjunto de graus de pertinência no intervalo $[0,1]$, que determina quantos grupos de pontos devem ser formados, podendo esses grupos serem denominados como níveis da região *plateau* do objeto.

Assim, para cada classe e para cada valor de grau de pertinência do nível previamente definido pelo usuário, forma-se um conjunto de todos os pontos com graus de pertinência

maiores do que 0, e com o grau de pertinência menor do que a distância entre o primeiro parâmetro (limite) e o grau de pertinência de cada nível da região *plateau* do objeto (segundo parâmetro). Conforme os autores, quanto maior o número de elementos do segundo parâmetro, mais suave é a transição entre os níveis da região *plateau* do objeto, e um ponto pode pertencer a vários grupos, pois estes podem estar dentro da distância entre o limite (primeiro parâmetro) e os diversos graus de pertinência dos níveis. É importante também que os valores do segundo parâmetro incluam todos os graus de pertinência das classes, para garantir que cada ponto pertença a pelo menos um conjunto (CARNIEL; SCHNEIDER, 2019).

Na próxima etapa, executa-se um algoritmo *convex hull* no conjunto para criar uma região que circunda minimamente a origem dos pontos. Os diversos *convex hull* gerados podem se sobrepor, pois cada ponto pode pertencer a mais de um nível. Entretanto, o *convex hull* junto com o seu grau de pertinência do segundo parâmetro forma a região *plateau* do objeto com um único componente, e é mesclado com a região *convex hull* final do objeto, que representa a classe, assim, aplica-se um operador *t-cnorma* para executar a operação de união e manter os graus de pertinências para as demais áreas.

2.4 Pacote FSR

O pacote *fsr*, desenvolvido na linguagem de programação R, foi apresentado por Carniel et al. (2021) como uma tecnologia para implementar tipos de dados espaciais *fuzzy*, operações e predicados da AEP, e uma solução para os problemas da construção espacial *fuzzy*. Até o momento do lançamento do pacote *fsr* não havia tecnologia que conectasse os SIG e os projetos de ciência de dados espaciais que tratassem a imprecisão dos fenômenos espaciais (CARNIEL et al., 2021).

O pacote *fsr* está disponível publicamente em <https://cran.r-project.org/package=fsr>, e ele é composto por quatro módulos que contém as funções para a implementação da AEP: o Módulo Básico, o Módulo de Construção, o Módulo de Manipulação de Dados Espaciais *Fuzzy*, e o Módulo de Inferência Espacial *Fuzzy*. Sua implementação consiste em outros pacotes do R, como o pacote R *Base*, para um suporte básico de programação do R, o pacote de *methods*, para definir as classes R e seus métodos formais, o pacote *sf*, para implementar tipos de dados espaciais *sfg* e manusear objetos espaciais *crisp* e o pacote *FuzzyR*, para criar as funções de pertinência das classes dos fenômenos *fuzzy*. Ainda, outros pacotes são também usado, como o *tidyverse* e *pso*.

Como um objeto espacial *plateau* é formado por uma lista de pares (componente), contendo um objeto espacial *crisp* e um grau de pertinência no intervalo de $[0,1]$, o pacote *fsr* fornece vários construtores para criar os componentes e os objetos geométricos. Exemplos de construtores foram apresentados no trabalho de (CARNIEL et al., 2021), como o `component_from_sfg`, e o `create_pgeometry`.

O construtor `component_from_sfg` recebe como entrada um objeto *sfg*, e um grau de pertinência, e produz como saída um componente. Já o construtor `create_pgeometry`,

aceita uma lista de componentes e um valor de *string* que indica o tipo de dado espacial *plateau* desejado, como exemplo, ponto *plateau*, linha *plateau* ou região *plateau*. Conforme Carniel et al. (2021), esses construtores do pacote *fsr* garantem que as propriedades da AEP sejam implementadas, pois o `component_from_sfg`, por exemplo, cria um objeto de componente apenas se objetos simples ou complexos de ponto, linha ou região for fornecido, e esse construtor não cria um objeto de componente se o objeto *sfg* não for do mesmo tipo de dado espacial simples ou complexo.

Utiliza-se também algumas funções do R para lidar com os componentes criados com esses construtores, como o método *show*, que é responsável por fornecer as representações textuais em PWKT do componente, ou o pacote *ggplot2*, para visualizar os componentes do objeto criado, que por padrão utiliza-se tons de cinza para indicar o quanto um ponto pertence a região *plateau* do objeto criado, na qual as cores mais escuras significam graus de pertinência maiores.

Já para a construção de objetos espaciais *plateau* com dados espaciais reais, a AEP define uma metodologia sistemática em duas etapas para a extração de dados, a etapa de *fuzzificação* e a etapa de construção, ambas com suas respectivas políticas, e o pacote *fsr* implementa essas etapas com a função `spa_creator`, conforme a expressão: `spa_creator <- function(tbl, fuzz_policy = "fsp", const_policy = "voronoi", ...)`.

O parâmetro *tbl* da função `spa_creator` é o conjunto de pontos espaciais com os seus valores numéricos. Já o parâmetro `fuzz_policy`, é para o usuário definir a política de *fuzzificação*, e o parâmetro `const_policy`, o usuário define a política de construção, entretanto até o momento, apenas a política de *Voronoi* e a política de Triangulação *Delanuay* foram implementadas na função `spa_creator`.

A política de *fuzzificação* padrão do `spa_creator` é a política de conjuntos *fuzzy*, a política padrão de construção é a política de *Voronoi*, e a saída do `spa_creator` é uma tabela contendo a região *plateau* do objeto formado. Entretanto, antes de chamar a função `spa_creator`, cria-se um vetor de *strings*, com as classes do fenômeno *fuzzy* da aplicação, e em seguida, define-se atribuindo a variáveis, as funções de pertinência das classes definidas anteriormente. Assim, chama-se a função `spa_creator`, para criar as regiões *plateau* do objeto, sendo que a primeira entrada é o vetor de *string* com os nomes das colunas, os vetores e as suas funções de pertinência, que são conectados pelo `spa_creator`.

3 RESULTADOS

3.1 Implementação do convex hull como uma nova política de construção

A política de construção *Convex hull* foi implementada no Programa 1 abaixo e conforme Carniel e Schneider (2019), essa política define dois parâmetros adicionais de entrada a serem definidos pelo usuário da aplicação, sendo o primeiro um limite para agrupar os pontos com graus de pertinência próximos de uma determinada classe de fenômenos *fuzzy*, sendo esse parâmetro um valor numérico entre $[0,1]$, e representado por *d* (Linha 1), e o segundo parâmetro é um vetor de graus de pertinência entre $[0,1]$, que determina quantos níveis devem ser formados na região *plateau*, para cada classe de fenômeno *fuzzy* da aplicação, sendo representado por *M* (Linha 1). Caso o usuário não defina esses parâmentros, definiu-se por padrão o limite de 0,05 para *d*, e uma sequência em um intervalo de 0,05, definida pela função *seq*, para o parâmentro *M*. Já o parâmetro *lp* (Linha 1), é o conjunto de dados espaciais com graus de pertinência, resultado da primeira fase de *fuzzificação*, e o parâmetro *base_poly*, é um objeto espacial de classe *sfg* utilizado para cortar os polígonos gerados, e por padrão deve ser vazio (Linha 1).

Após a definição dos primeiros parâmetros, cria-se uma lista vazia, *result_classes* (Linha 2), com o tamanho das classes de *lp*, que representa os fenômenos espaciais *fuzzy* da aplicação, para armazenar posteriormente as regiões *plateau* formadas para cada classe. Na Linha 3, *cls* recebe o nome das classes de *lp*, para ser acoplado posteriormente na tabela de saída do algoritmo, e nomear as regiões *plateau* formadas.

Para cada classe de fenômenos *fuzzy* das coluna de *lp* (Linha 5), e para cada grau de pertinência de *M* (Linha 8), o algoritmo atribui em *res* (Linha 9), um conjunto com todos os pontos com grau de pertinência maior do que 0 e com o grau de pertinência menor do que a distância entre o limite *d*, e o grau de pertinência de cada nível de *M*. Para a formação de um polígono (região), são necessários o mínimo de 3 pontos, assim, o algoritmo verifica se para cada classe, o número de linhas (pontos selecionados) é maior do que 2 (Linha 11). Para os casos verdadeiros, em *pts* (Linha 12), utiliza-se a função *st_as_sf*, do pacote *sf*, para converter *res* em um objeto espacial, e em *ch* (Linha 13), utiliza-se a função *st_geometry*, do pacote *sf*, para definir a geometria espacial de *pts*, e aplicar a operação espacial *convex_hull*, através da função *st_convex_hull*, também do pacote *sf*.

O algoritmo verifica se *ch* é uma região espacial (Linha 15), e se o objeto espacial *base_poly* é vazio (Linha 17). Assim, *ch* receberá a intersecção entre *ch* e *base_poly* (Linha 18), e em *comp* (Linha 21), utiliza-se o construtor *componente_from_sfg*, do pacote *fsr*, para criar os componentes de *ch*, para cada nível de *M*. Em *pregion* (Linha 22), utiliza-se o construtor *create_pgeometry*, para receber a lista de componentes de *ch*, como uma região *plateau*.

Como cada ponto de *lp* pode pertencer a mais de um nível, os diversos *convex hull*

Programa 1: Política de Construção *Convex hull*.

```
1  convex_hull_policy <- function(lp, M = seq(0.05, 1, by = 0.05), d = 0.05,
   base_poly = NULL, ...) {
2   result_classes <- list(ncol(lp) - 3)
3   cls <- colnames(lp)[-c(1:3)]
4
5   for(k in 4:ncol(lp)) {
6     result_classes[[k-3]] <- create_empty_pgeometry("PLATEAUREGION")
7
8     for (level in M){
9       res <- lp %>% filter((!!as.symbol(cls[k-3])) > 0 &
   (abs((!!as.symbol(cls[k-3])) - level) <= d))
10
11      if(nrow(res) > 2) {
12       pts <- st_as_sf(res, coords = c(1, 2))
13       ch <- st_convex_hull(do.call(c, st_geometry(pts)))
14
15        if(inherits(ch, c("POLYGON", "MULTIPOLYGON"))){
16
17          if(!is.null(base_poly) && inherits(base_poly, c("POLYGON",
   "MULTIPOLYGON"))) {
18           ch <- st_intersection(ch, base_poly)
19          }
20
21          comp <- component_from_sfg(ch, level)
22          pregion <- create_pgeometry(list(comp), "PLATEAUREGION")
23          result_classes[[k-3]] <- spa_union(result_classes[[k-3]],
   pregion)
24         }
25       }
26     }
27   }
28
29   pgeoms <- tibble(class = cls, pgeometry = result_classes)
30 }
```

gerados podem se sobrepor, assim, *result_classes* receberá a união (Linha 23) das regiões *plateau* de cada classe dos fenômenos *fuzzy* de *pregion*. Conforme Carniel e Schneider (2019), a saída dos algoritmos da etapa de construção consistem na região *plateau* criada e a sua classe associada, com isso, atribui-se em *pgeoms* (Linha 28) um *tibble*, sendo a primeira coluna com os nomes das classes de *lp* e a segunda coluna com as regiões *plateau* formadas.

3.2 Acoplando a política na função spa_creator

O pacote *fsr* implementa as etapas de *fuzzificação* e construção da AEP através da função *spa_creator* (CARNIEL; SCHNEIDER, 2019), assim, para incluir o Programa 1 como um argumento dessa função, é necessário incluir o *script* do algoritmo no arquivo *builder_functions.R*, do código fonte do pacote *fsr*, disponível em <https://github.com/accarniel/fsr>.

Apresentou-se para os autores do pacote *fsr* uma primeira versão do *script* como uma *Issue* do Github, disponível em <https://github.com/accarniel/fsr/issues/29>. Realizou-

se algumas adaptações desssa primeira versão do *script*, para deixá-la mais genérica, antes de implementá-la no código fonte, como por exemplo, a definição dos valores padrões dos parâmetros *M* e *d* (Linha 1), a alteração do parâmetro *fsp*, para *lp* (Linha 1), como padrão do código fonte do pacote *fsr*, a inclusão da validação do número de linhas para a formação do polígono (Linha 11), entre outras alterações.

Com isso, ao incluir o Programa 1 no código fonte do pacote *fsr*, a função *spa_creator* receberá no parâmetro *const_policy*, um novo argumento, o *convex_hull*, que receberá o conjunto de dados espaciais *fuzzificado*, da etapa de *fuzzificação*, como primeiro parâmetro, e executará a política de construção *Convex hull*.

3.3 Estudo de Caso

Após a implementação do Programa 1 no código fonte do pacote *fsr*, faz-se necessário a validação do *script* em relação ao seu funcionamento, e isso foi realizado através de um estudo de caso registrado no Programa 2. Nesse estudo de caso, simulou-se uma base de dados espaciais e 4 classes do fenômeno temperatura, assim, executou-se a função *spa_creator* variando as políticas de construção *Convex hull*, *Voronoi* e a triangulação *Delaunay*, para analisar as regiões *plateau* formadas.

Antes de iniciar a criação da base de dados espacial e as classes do fenômeno *fuzzy*, é necessário habilitar alguns pacotes do R (Linhas 1 a 5). Para a base de dados espacial, criou-se na variável *tbl* 100 pontos para as coordenadas *x*, entre [0, 30], e *y*, entre [0,50], e também as temperaturas em *z*, variando entre [-50,200] (Linha 8). Com isso, criou-se as classes *super_cold*, *cold*, *hot* e *super_hot* (Linha 9), e as suas funções de pertinência trapezoidal para a classe *cold*, e triangular para as demais classes (Linhas 10 a 13). Para a criação das regiões *plateau* utilizando a política de construção *Convex hull*, criou-se a variável *result_convex_hull* e atribuiu a essa variável a função *spa_creator*, com os parâmetros *tbl*, os nomes das classes criadas atribuídas na variável *classes*, as funções de pertinências criadas, atribuídas como um vetor na variável *mfs*, a política de *fuzzificação* como *fsp* (política de conjuntos *fuzzy*), e a política de *convex_hull* em *const_policy* (Linha 15). Definiu-se 3 níveis com os graus de pertinência de 0,8, 0,5 e 0,2, atribuidos como um vetor em *M*, e o limite para as regiões *plateau* de 0.3, atribuído em *d* (Linha 15).

A saída das políticas de construção (saída do algoritmo) são as tabelas contendo uma coluna com os nomes das classes das regiões *plateau* e a outra coluna com as representações textuais das regiões *plateau* formadas, junto com os graus de pertinência. Assim, para a visualização das regiões *plateau* formadas a partir da política de construção *Convex hull*, utilizou-se a função *plot* para visualizar as colunas *pgeometry*, que contém as representações textuais das classes das regiões *plateau* formadas (Linhas 16 a 19).

Observa-se na Figura 5, as regiões *plateau* formadas para cada classe, *super_cold*, *cold*, *hot* e *super_hot*, e os polígonos de *convex hull* para cada um dos possíveis níveis formados, e quanto mais escuro o tom de cinza, maior o grau de pertinência dos pontos para a classe, ou

Programa 2: Estudo de caso: contrução de regiões *plateau* com spa_creator

```
1  library(devtools)
2  library(fsr)
3  library(tidyverse)
4  library(sf)
5  library(FuzzyR)
6
7  set.seed(7)
8  tbl = tibble(x = runif(100, min= 0, max = 30), y = runif(100, min = 0, max
     = 50), z = runif(100, min = -50, max = 200))
9  classes <- c("super_cold", "cold", "hot", "super_hot")
10 cold_mf <- genmf("trapmf", c(0, 10, 20, 35))
11 hot_mf <- genmf("trimf", c(35, 50, 100))
12 super_cold_mf <- genmf("trimf", c(-50, -25, 0))
13 super_hot_mf <- genmf("trimf", c(100, 150, 200))
14
15 result_convex_hull <- spa_creator(tbl, classes = classes, mfs =
     c(super_cold_mf, cold_mf, hot_mf, super_hot_mf), fuzz_policy = "fsp",
     const_policy = "convex_hull", M = c(0.8, 0.5, 0.2), d = 0.3)
16 plot(result_convex_hull$pgeometry[[1]])
17 plot(result_convex_hull$pgeometry[[2]])
18 plot(result_convex_hull$pgeometry[[3]])
19 plot(result_convex_hull$pgeometry[[4]])
20
21 result_voronoi <- spa_creator(tbl, classes = classes, mfs =
     c(super_cold_mf, cold_mf, hot_mf, super_hot_mf))
22 plot(result_voronoi$pgeometry[[1]])
23 plot(result_voronoi$pgeometry[[2]])
24 plot(result_voronoi$pgeometry[[3]])
25 plot(result_voronoi$pgeometry[[4]])
26
27 result_delaunay <- spa_creator(tbl, classes = classes, mfs =
     c(super_cold_mf, cold_mf, hot_mf, super_hot_mf), fuzz_policy = "fsp",
     const_policy = "delaunay")
28 plot(result_delaunay$pgeometry[[2]])
29 plot(result_delaunay$pgeometry[[3]])
30 plot(result_delaunay$pgeometry[[4]])
```

seja, o grau de pertinência é mais próximo de 1. Observa-se que para as classes *super_cold* e *super_hot*, formou-se os polígonos de *convex hull* para os três níveis, pois o resultado computado para essas classes e para cada nível, conteve pontos com grau de pertinência maiores do que 0 e menores do que a distância entre o limite d e o grau de pertinência de M.

Para a criação das regiões *plateau* através da política de construção *Voronoi*, criou-se a variável *result_voronoi*, e utilizou-se as configurações padrões do *spa_creator* (Linha 21), e a função *plot* para a visualização das reigiões *plateau* formadas (Linhas 22 a 25). Observa-se na Figura 6, que formou-se células *Voronoi* para as quatro classes, pois obteve-se para todas as classes, pontos com graus de pertinência maiores do que 0.

Figura 5 – Política de Construção *Convex hull* e as Regiões *Plateau* por classe

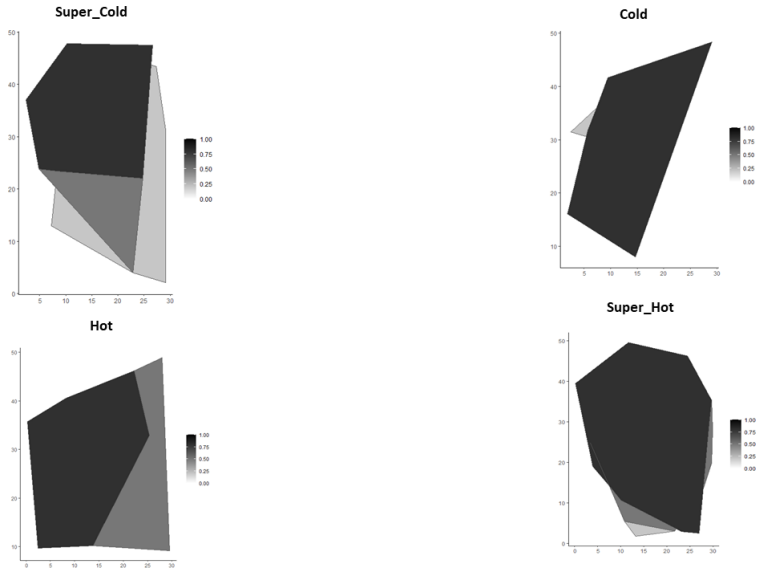
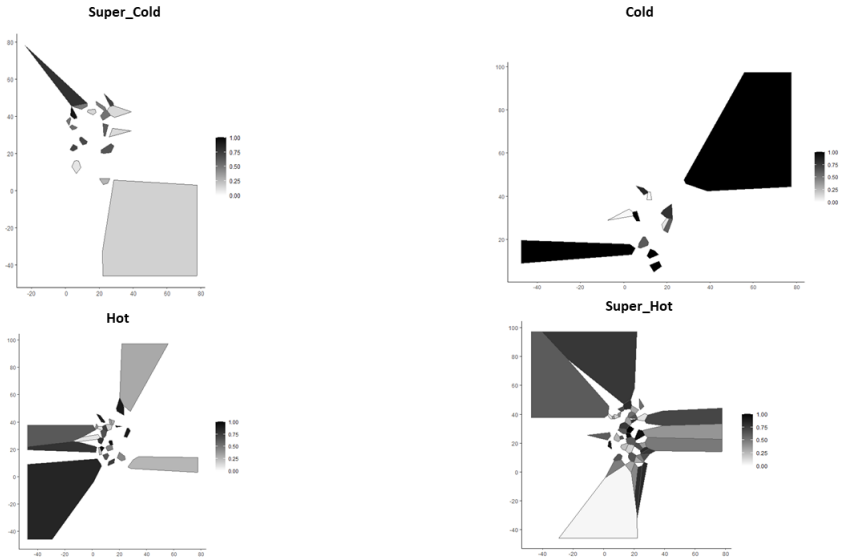
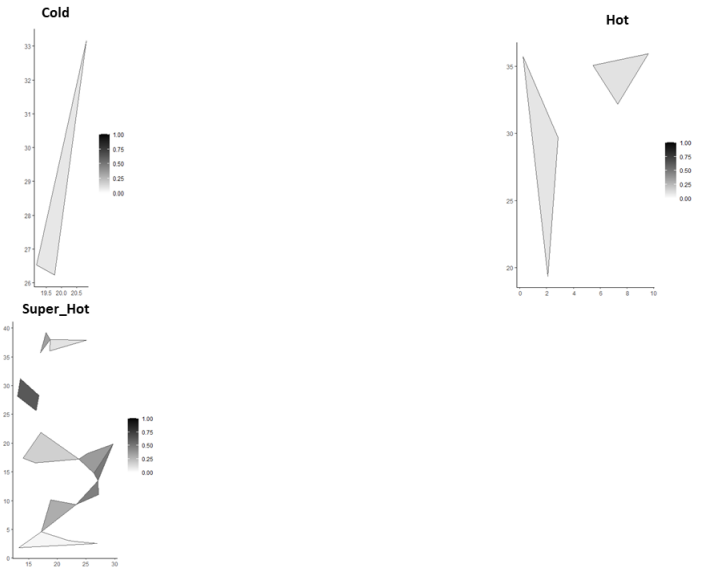


Figura 6 – Política de Construção *Voronoi* e as Regiões *Plateau* por classe



Já para a criação das regiões *plateau*, através da política de construção de Triangulação *Delaunay*, criou-se a variável *result_delaunay*, e atribuiu a essa variável, a função *spa_creator*, com a política de *fuzzificação* de conjuntos *fuzzy* (*fsp*) (Linha 27). Para a visualização das triangulações formadas para cada região *plateau* formada, utilizou-se também a função *plot* (Linhas 28 a 30). Observa-se na Figura 7, que para a classe *super_cold* formou-se uma região *plateau* vazia, pois para essa classe, não obteve-se pontos com grau de pertinência maiores do que 0.

Figura 7 – Política de Construção Triangulação *Delaunay* e as Regiões *Plateau* por classe



4 CONCLUSÃO

Este trabalho implementou a política de construção *Convex hull* no pacote *fsr* através do desenvolvimento do algoritmo na linguagem de programação R, que até o momento não havia sido implementada. Através da simulação da base de dados espaciais e das quatro classes criadas de conjuntos *fuzzy* do fenômeno temperatura, pode-se validar progressivamente se as operações e os conceitos da AEP foram corretamente implementados.

Assim, observou-se que o algoritmo atendeu aos conceitos da AEP, ao criar para cada classe de fenêmenos *fuzzy* da base de dados simulada, os componentes, as regiões *plateau* para as classes, e aplicou as regras da política de construção *Convex hull*, conforme descrita no trabalho de (CARNIEL; SCHNEIDER, 2019). Pode-se também observar através das imagens geradas no estudo de caso, as diferenças entre as políticas de contrução *Convex hull*, *Voronoi* e Triangulação *Delaunay*. Isso mostra a importância da política de construção *Convex hull*, pois o usuário consegue observar através dos polígonos de *convex hull*, todos os pontos espaciais que pertencem a cada nível e a cada região *plateau* da classe analisada.

Para trabalhos futuros, recomenda-se aplicar o algoritmo desenvolvido em uma base de dados espaciais reais bem como uma análise mais detalhada sobre as diferenças entre as políticas de construção. Por fim, há também o planejamento de realizar estudos empíricos que estudem o tempo de execução das políticas de construção.

Referências

CARNIEL, A. C. Spatial information retrieval in digital ecosystems: A comprehensive survey. In: **Proceedings of the 12th International Conference on Management of Digital EcoSystems**. [S.l.: s.n.], 2020. p. 10–17. Citado 2 vezes nas páginas 13 e 14.

CARNIEL, A. C. et al. Handling fuzzy spatial data in r using the fsr package. In: **Proceedings of the 29th International Conference on Advances in Geographic Information Systems**. [S.l.: s.n.], 2021. p. 526–535. Citado 3 vezes nas páginas 10, 18 e 19.

CARNIEL, A. C.; SCHNEIDER, M. A conceptual model of fuzzy topological relationships for fuzzy regions. In: IEEE. **2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)**. [S.l.], 2016. p. 2271–2278. Citado na página 9.

CARNIEL, A. C.; SCHNEIDER, M. Spatial plateau algebra: An executable type system for fuzzy spatial data types. In: **2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)**. [S.l.: s.n.], 2018. p. 1–8. Citado 2 vezes nas páginas 9 e 15.

CARNIEL, A. C.; SCHNEIDER, M. A systematic approach to creating fuzzy region objects from real spatial data sets. In: IEEE. **2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)**. [S.l.], 2019. p. 1–6. Citado 8 vezes nas páginas 9, 10, 16, 17, 18, 20, 21 e 26.

CHEN, C.; RAZAK, T. R.; GARIBALDI, J. M. Fuzzyr: An extended fuzzy logic toolbox for the r programming language. In: IEEE. **2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)**. [S.l.], 2020. p. 1–8. Citado 2 vezes nas páginas 9 e 11.

DILO, A.; de By, R. A.; STEIN, A. A system of types and operators for handling vague spatial objects. **International Journal of Geographical Information Science**, v. 21, n. 4, p. 397–426, 2007. Citado na página 9.

GÜTING, R. H. An introduction to spatial database systems. **the VLDB Journal**, Springer, v. 3, n. 4, p. 357–399, 1994. Citado 3 vezes nas páginas 9, 13 e 14.

SCHNEIDER, M.; BEHR, T. Topological relationships between complex spatial objects. **ACM Transactions on Database Systems (TODS)**, ACM New York, NY, USA, v. 31, n. 1, p. 39–81, 2006. Citado 2 vezes nas páginas 13 e 14.

ZADEH, L. Fuzzy sets. **Information and Control**, v. 8, n. 3, p. 338–353, 1965. ISSN 0019-9958. Disponível em: <https://www.sciencedirect.com/science/article/pii/S001999586590241X>. Citado 2 vezes nas páginas 9 e 11.

ZADEH, L. A. Fuzzy logic—a personal perspective. **Fuzzy sets and systems**, Elsevier, v. 281, p. 4–20, 2015. Citado na página 11.

ZIMMERMANN, H.-J. Fuzzy set theory. **Wiley Interdisciplinary Reviews: Computational Statistics**, Wiley Online Library, v. 2, n. 3, p. 317–332, 2010. Citado 2 vezes nas páginas 9 e 11.