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

CLAISA YATHIE LUBKE

# IMPLEMENTAÇÃO DE CONSULTAS ESPACIAIS BASEADAS EM DISTÂNCIA USANDO A LINGUAGEM DE PROGRAMAÇÃO R

TRABALHO DE CONCLUSÃO DE CURSO DE ESPECIALIZAÇÃO

DOIS VIZINHOS
2022

CLAISA YATHIE LUBKE

# IMPLEMENTAÇÃO DE CONSULTAS ESPACIAIS BASEADAS EM DISTÂNCIA USANDO A LINGUAGEM DE PROGRAMAÇÃO R

# IMPLEMENTATION OF SPATIAL QUERIES BASED ON DISTANCE USING THE R PROGRAMMING LANGUAGE

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. Rafael Alves de Oliveira

DOIS VIZINHOS
2022

CLAISA YATHIE LUBKE

# IMPLEMENTAÇÃO DE CONSULTAS ESPACIAIS BASEADAS EM DISTÂNCIA USANDO A LINGUAGEM DE PROGRAMAÇÃO R

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: 11/novembro/2022

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

Yuri Kazubowski Lopes
Doutorado
Universidade do Estado de Santa Catarina

Francisco Carlos Monteiro Souza
Doutorado
Universidade Tecnológica Federal do Paraná - Câmpus Dois Vizinhos

DOIS VIZINHOS
2022

## AGRADECIMENTOS

Quero agradecer de maneira especial ao professor Anderson Chaves Carniel pela disponibilidade, senso crítico e apoio neste curso. Certamente sem seu compromisso, dedicação e real orientação, a realização desse trabalho não seria possível.

Fica aqui também registrada a minha gratidão a Universidade Tecnológica Federal do Paraná, ao pessoal da secretaria e aos professores do Curso de Especialização em Ciência de Dados por todo suporte e conhecimento compartilhado durante esse processo.

Finalizo agradecendo aos meus familiares, e aos meus amigos pessoais, principalmente o Luiz, pelo apoio que foi muito importante nessa etapa.

## RESUMO

Em banco de dados espaciais, a forma como um dado geométrico se relaciona com outro dado geométrico é o relacionamento espacial, cujo pode ser de três principais tipos: topológico, métrico ou direcional. Cada relacionamento possui as suas características próprias para a elaboração das suas consultas espaciais. Definindo-se que a distância entre os objetos sempre se dá em linha reta e que existem diversas formas de interação entre objetos espaciais baseados em distância, é apresentado neste trabalho a implementação de consultas espaciais baseadas em distância com a linguagem R, baseadas na função k-NN. Desta forma, é possível padronizar tais consultas e promover agilidade na sua utilização, pois são complexas a sua elaboração.


**Palavras-chave**: k-NN; consulta espacial; linguagem R.

## ABSTRACT

In spatial databases, the way in which a geometric data relates to another geometric data is the spatial relationship, which can be of three main types: topological, metric or directional. Each relationship has its own characteristics for the elaboration of its spatial queries. Defining that the distance between objects is always in a straight line and that there are several forms of interaction between spacious objects supported in distance, this work presents the implementation of spiritual consultations registered in distance with the R language, inspired by the function k-NN. In this way, it is possible to standardize such queries and promote agility in their use, as their elaboration is complex.



**Keywords**: k-NN; spatial query; R language.

SUMÁRIO


1 INTRODUÇÃO . . . . . 7
1.1 Justificativa . . . . . 7
1.2 Objetivos . . . . . 8
1.3 Contribuições . . . . . 8
1.4 Organização do Trabalho . . . . . 8

2 FUNDAMENTAÇÃO TEÓRICA . . . . . 9
2.1 Banco de dados espaciais . . . . . 9
2.2 Consultas espaciais . . . . . 9

3 TRABALHOS RELACIONADOS . . . . . 11
3.1 PostGIS . . . . . 11
3.2 Pacote sf . . . . . 12
3.3 Seleção de consultas . . . . . 13

4 IMPLEMENTAÇÃO DE CONSULTAS ESPACIAIS BASEADAS EM DISTÂNCIA . . . . . 14
4.1 Base de dados para exemplos . . . . . 14
4.2 k-NN . . . . . 16
4.3 Range Query . . . . . 18
4.4 k-NN + Range Query . . . . . 21
4.5 Relative k-NN . . . . . 22
4.6 Reverse k-NN . . . . . 24
4.7 k Closest Pairs . . . . . 27

5 CONCLUSÃO . . . . . 32

REFERÊNCIAS . . . . . 33

# 1 INTRODUÇÃO

Na modelagem de dados existe uma etapa chamada abstração, que a descrição da realidade. Nesse processo, quando fazemos a abstração do mundo real não significa que devemos descreve-lo exatamente como ele é, podemos levar em consideração algumas características aproximadas, outras simplificadas e outras podem ser ignoradas. Isso faz parte da base de coleta de dados. Quando a abstração descreve os objetos do mundo real podemos obter os dados conhecidos como espaciais (MARINO).

Assim como os dados convencionais são manipulados num banco de dados, existe a possibilidade de incluir os dados espaciais em um banco de dados também, quando isso é possível, temos o banco de dados espacial.

Banco de dados espaciais são banco de dados que suportam, além dos tipos de dados convencionais, os dados espaciais, os quais podem ser de dois tipos primários: vetoriais ou matriciais. Os dados vetoriais são referentes a descrever a posição ou a dimensão (extensão ou área) dos objetos do mundo real, já os matriciais, representados por matrizes, estão vinculado a imagens (MARINO).

Quando realizamos consultas envolvendo os dados espaciais, temos as consultas espaciais. A relevância em consultas espaciais podem ser aplicadas a várias áreas. Exemplos:

- numa busca de escolas mais próximas a uma determinada localização;
- distribuição tática de soldados;
- análise de desmatamento de uma floresta;
- características dos tipos de solo numa região.

As consultas espaciais podem ser classificadas fundamentalmente como topológicas, baseadas em direção ou baseadas em distância, dependendo do seu objetivo. Além destas classificações podem haver a mistura entre eles, resultando em consultas espaciais híbridas.

Neste trabalho é utilizado os dados espaciais vetoriais (os quais podem ser representados por ponto, linha ou polígono, fundamentalmente) (GÜTING, 1994). Com isso, são desenvolvidas consultas espaciais baseadas em distância. A ideia é facilitar o acesso a estes tipos de consultas, devido à complexidade em elaborá-las. Assim, estas consultas podem ser utilizadas por diferentes usuários, promovendo-lhes agilidade em suas pesquisas, análises e tomadas de decisão. A linguagem R possui bibliotecas que permitem a manipulação destes dados, por exemplo a *sf* (PEBESMA).

## 1.1 Justificativa

As consultas espaciais requerem um projeto de implementação complexa que envolvem um conhecimento especializado de recursos existentes. Devido a estas características, desenvolver algo que facilite as consultas espaciais aos usuários promoveria agilidade e padronização.

## 1.2 Objetivos

O objetivo geral é a implementação de consultas espaciais com a linguagem R que possibilite a execução de consultas espaciais baseadas em distância. Os objetivos específicos são:

- pesquisar funções que se enquadrem como consultas espaciais baseadas em distância para incluir no pacote;
- implementar as consultas selecionadas através da linguagem R;
- elaborar exemplos de utilização das consultas.

## 1.3 Contribuições

Este trabalho apresenta as seguintes contribuições:

- seleção de consultas espaciais baseadas em distância (função k-NN e suas variantes);
- desenvolvimento dos algoritmos das funções selecionadas, utilizando a linguagem R com o auxílio da bibioteca *sf* (para manipular dados espaciais);
- apresentação dos resultados das consultas desenvolvidas numa base de dados de testes.

## 1.4 Organização do Trabalho

Este trabalho é estruturado da seguinte forma, a partir dos próximos capítulos: no Capítulo 2 é feita a fundamentação teórica, onde é explicado sobre banco de dados espaciais e consultas espaciais. A seguir, no Capítulo 3, são apresentados dois trabalhos relacionados com consultas espaciais, e apresenta-se o PostGIS e a biblioteca sf. No Capítulo 4 são mostradas as consultas desenvolvidas com exemplo aplicado numa base de teste. Por fim, no Capítulo 5 é feita a conclusão deste trabalho e observações para trabalhos futuros.

# 2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção é apresentado o que são banco de dados espaciais, consultas espaciais baseadas em distância.

## 2.1 Banco de dados espaciais

Se tratando de banco de dados, houve a necessidade de manipulação de dados espaciais em Sistemas de Banco de Dados (SGBD). Com o tempo houveram diversos estudos em como incorporar os dados espaciais nestes sistemas. Hoje, existe a possibilidade de utilizar dados espaciais num SGBD, através da ferramenta chamada de Sistemas de Banco de Dados Espaciais (SGBDE). Um banco de dados espacial é um banco de dados convencional que possui uma extensão, a qual oferece tipos de dados espaciais em seu modelo de dados e linguagem de consulta para estes tipos, e provê a possibilidade de indexação espacial (GÜTING, 1994).

Um Sistema de Informação Geográfica (SIG) é um sistema capaz de lidar com dados convencionais e dados espaciais. CÂMARA et. al. aponta duas principais caracteristicas de um SIG: a possibilidade de integração de diferentes dados espaciais em uma mesma base de dados, e; oferecer formas de recuperação, manipulação e visualização destes dados. Um SIG contém um banco de dados espaciais, então um SBDE é um SIG.

Em um SIG, os dados espaciais suportados por ele podem representar um fato, fenômeno e localização de um objeto espacial. Considerando que o objeto está sobre a superfície da Terra, podemos chamar esses dados de geográficos ou georreferenciados (CÂMARA et al.). Os objetos espaciais (geográficos) podem ser representados de forma vetorial, que seria o dado vetorial.

Os dados vetoriais abrangem os tipos de dados espaciais que representam a posição de objetos no espaço. Estes objetos possuem três tipos fundamentais de abstrações: ponto, linha e polígono. Um ponto é a representação de um objeto espacial , onde a importância é apenas a sua localização, desconsiderando as dimensões do objeto. Por exemplo, indicar num mapa a localização de restaurantes, estes são representados por pontos. Uma linha representa objetos espaciais que sua extensão é relevante, é uma sequencia de pontos que se ligam. Ela pode representar um rio, por exemplo. E um polígono representa objetos espaciais que tem a área relevante (GÜTING, 1994).

## 2.2 Consultas espaciais

A interação entre objetos espaciais é chamada de relacionamento espacial, cujo pode ser dividido em classes baseado-se na topologia, na distância ou na direção dos objetos. Estas classificações possuem consultas espaciais caracteríticas. Destre elas, as consultas espaciais baseadas em distância (CARNIEL, 2020).

As consultas espaciais baseadas em distância (ou métricas) são caracterizadas nas distâncias entre os objetos. CARNIEL (2020) fala em três tipos de consultas espaciais baseadas em distância: com operadores unários, com operadoes binários e recuperação de vizinhos mais próximos.

As consultas espaciais baseadas em distância com operadores unários são caracterizadas pela recuperação de um valor, por exemplo, para saber o comprimento de uma linha, ou a área de um polígono. Os operadores binários permitem a recuperação de objetos espaciais dado uma distância, ou um intervalo. E as recuperações dos vizinhos mais próximos são caracterizadas por recuperar os objetos mais próximos a um objeto determinado (CARNIEL, 2020). Este trabalho traz as seguintes consultas espaciais baseadas em distância de recuperação de vizinhos mais próximos:

- k-NN - significa k Nearest neighbour (k Vizinhos mais próximos). Este algoritmo retorna os vizinhos mais próximos a uma referência. Do ponto de vista de banco de dados espacial, os vizinhos mais próximos a um objeto de referência (ponto de consulta) encontram os k objetos mais próximos deste ponto de consulta com base em sua distância espacial. O valor de k faz referência a quantidade de vizinhos que devem ser buscados.
- Range Query - este algoritmo retorna os vizinhos mais próximos a um objeto de referência com base numa distância máxima entre o objeto de referência e os vizinhos. Os vizinhos encontrados precisam ter a distância igual ou menor a distância máxima que é fornecida.
- k-NN + Range Query - este algoritmo, como o nome sugere, une os algoritmos kNN e Range Query. Na primeira etapa,o algoritmo executa o kNN, o resultado dessa busca se torna a nova vizinhança do ponto de busca. Na próxima etapa, desta nova vizinhança retornada no kNN são buscados os vizinhos que atendem a distância máxima fornecida na Range Query
- Relative k-NN - este algoritmo utiliza o k-NN a Range Query na sua estrutura. Na primeira etapa, é executado o k-NN, deste resultado é armazenado a distância do k item mais distante. Depois, sobre essa distância armazenada é acrescido um porcentagem (a qual é fornecida na chamada do algoritmo). Por fim, é executada a Range Query, onde a distância máxima é esta distância acrescida de uma porcentagem.
- Reverse k-NN - este algoritmo executa o k-NN para todos os pontos da vizinhança. Se o ponto de referência está dentre os k itens do k-NN do ponto da vizinhança, então este ponto da vizinhança é um dos resultados desse algoritmo. Dessa forma, é verificado se o ponto de referência está contido em cada k-NN realizado nos pontos da vizinhança, caso esteja, o ponto vai fazer parte do resultado.
- k Closest Pairs - este algoritmo se difere dos anteriores, quando se trata do ponto de referência. Aqui, o ponto de referência deve ser uma base de dados com diversos pontos, assim como a vizinhança, não apenas um ponto. Ele funciona calculando o k-NN de cada ponto da base de referência, com cada ponto da base da vizinhança. Depois, são retornados o k pares que possuem a menor distância entre eles.

# 3 TRABALHOS RELACIONADOS

Neste capítulo é apresentado o PostGIS e a biblioteca sf do R, e como foram escolhidas as consultas implementadas neste trabalho.

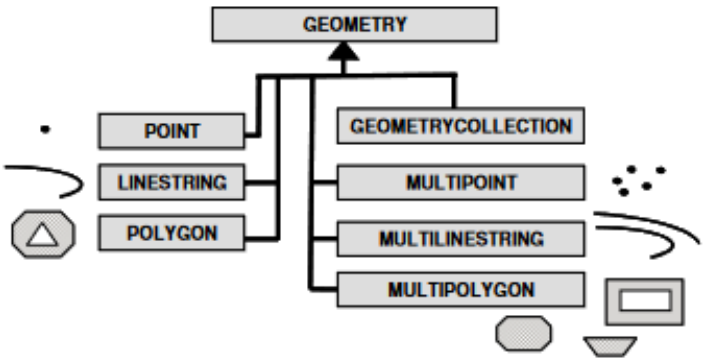
## 3.1 PostGIS

O PostGIS é uma extensão de banco de dados espaciais do PstgresSQL. É uma ferramenta robusta para manipulação de dados espaciais. Possui indexação espacial baseada no R-Tree (GIST) e funções para análise e processamento de objetos espaciais (CASANOVA et al.).

O tipo de dado espacial existente no PostGIS é o *geometry*. Porém, ele tem subclasses, possibilitando o usuário fixar que o tipo de dados que uma coluna pode receber, ao se definir na criação de uma tabela, em apenas dados geométricos do tipo ponto, por exemplo. A Figura 1 mostra a hierarquia que o dado *geometry* possui: os tipos de dados *point*, *linestring*, *polygon*, *multipoint*, *multilinestring*, *multipolygon* e *geometrycollection* são compreendidos como um dado *geometry* (CASANOVA et al.).

O PostGIS implementa padrões da o *Open Geospatial Consortium* (OGC), que é um consórcio que desenvolve padrões para facilitar a comunicação entre sistemas diferentes e engloba também informações espaciais, OpenGIS (DAVIS JR et al.). O OpenGIS possui um conceito abstrato chamado simple feature, que está associado ao armazenamento de dados geográficos nas tabelas (DAVIS JR et al.).

Figura 1 – Hierarquia do dado espacial *geometry*




Fonte: adaptado de CASANOVA et al.

Como o PostGIS tem funções próprias para os dados espaciais, é possível realizar consultas espaciais neste sistema. Quando tratamos de consultas espaciais, há um fator muito

importante que precisa ser levado em consideração, o sistema de referência espacial. Aqui no PostGIS ele chamado de SRID (Spatial Reference IDentifier). O SRID descreve os dados espaciais de uma determinada região do planeta Terra, causando uma menor distorção entre estes dados. Não é possível uma consulta espaciais caso os dados espaciais tenham um SRID diferente.

O Listing 3.1 mostra o exemplo da função espacial st_distance, que calcula a menor distância entre dois objetos espaciais num plano 2D, e da função espacial st_transform, que possibilita mudar o sistema de referência espacial sobre a coordenada que foi passada. No Listing 3.1 é mostrado a consulta com o SRID 4326, o qual retorna o resultado em graus, e a consulta com o SRID 3857 que calcula a distância entre os pontos em metros (POSTGIS).

```
1 SELECT
2     st_distance ('SRID=4326; POINT(10 5)', 'SRID=4326; POINT(23 13)') as
      graus,
3     st_distance(st_transform('SRID=4326; POINT(10 5)', 3857),
4         (st_transform('SRID=4326; POINT(23 13)', 3857)) as metros;
```

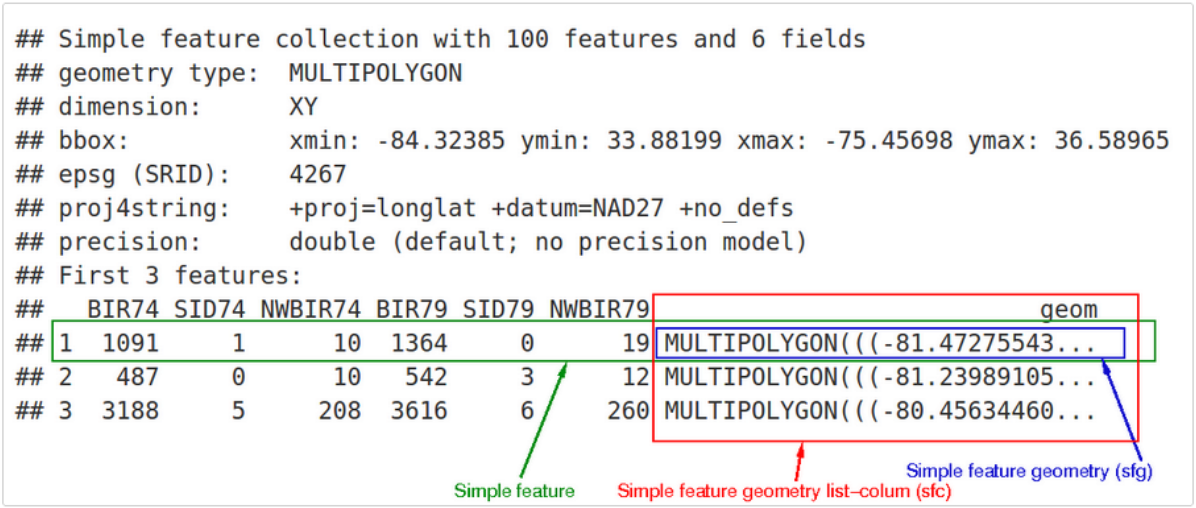
Listing 3.1 – Exemplo de consulta espacial com o PostGIS.

## 3.2 Pacote sf

O nome da biblioteca *sf* vem de simple feature, que é definido pela OGC. Simple feature descrevem como um objeto do mundo real pode ser representado no computador. E eles são organizados da seguinte forma (Figura 2) (PEBESMA):

- *sf* - é a tabela contendo característica geométricas e convencionais;
- *sfc* - é a tabela contendo apenas a coluna geométrica;
- *sfg* - é a o valor de uma célula da coluna geométrica.

Figura 2 – Representação de *sf*, *sfc* e *sfg* na tabela




Fonte: adaptado de PEBESMA

Devido o formato do planeta Terra ser irregular, se tratando de dados espaciais, existe a necessidade de que cada parte do planeta possua algo que o caracterize, para tornar os seus objetos espaciais com uma menor distorção ao representa-los computacionalmente. Para isso existe o CRS (*Coordinate reference systems*), que é um código que muda dependendo da região do planeta, que como dito anteriormente, reduz as distorções numa representação vetorial. Então, num relacionamento espacial, é importante que o CRS dos dados envolvidos sejam o mesmo, senão, não é possivel realizar este relacionamento (PEBESMA). O CRS aqui no *sf* equivale ao SRID no PostGIS.

A biblioteca *sf* fornece ferramentas para a manipulação de dados espaciais no R, disponibilizando algumas funções que permitem a utilização de dados espaciais em consultas espaciais elaboradas pelo usuário.

## 3.3 Seleção de consultas

O artigo *A taxonomy for nearest neighbour queries in spatial databases* (TANIAR; RAHAYU, 2013) apresenta conceitos de diversas consultas espaciais, com alguns exemplos envolvendo pontos, para o kNN e suas variações. A partir dele selecionou-se a maioria das consultas desenvolvidas neste trabalho. E o artigo *Closest Pair Queries in Spatial Databases* (CORRAL et al., 2000) apresenta a ideia do que seria a consulta dos pares mais próximos e com isso foi feita a versão apresentada neste trabalho. Para todas as consultas é considerada a distância euclidiana

# 4 IMPLEMENTAÇÃO DE CONSULTAS ESPACIAIS BASEADAS EM DISTÃNCIA

Este capítulo apresenta as consultas desenvolvidas e exemplos de suas aplicações sobre uma base de teste.

As funções apresentadas neste capítulo, que implementam as consultas espaciais baseadas em distância, são disponibilizadas no pacote sqr, que está em construção, mas pode ser acessado pelo link https://github.com/accarniel/sqr.

## 4.1 Base de dados para exemplos

As consultas espaciais baseadas em distância, neste trabalho, são exemplificadas com dados geométricos do tipo ponto, para uma compreensão mais simples do funcionamento de cada consulta visualmente. Então, nas próximas seções deste capítulo, escolheu-se uma base no Open Street Map (OSM), através da biblioteca 'osmdata', as características dessa base é que são dados geométricos do tipo ponto de um grupo de *aeroway*, subgrupo *windsock* , na região do Paraná (Listing 4.1).

```
1 library("osmdata")
2 q <- opq(bbox = 'parana brazil') %>%
3   add_osm_feature(key='aeroway', value='windsock') %>%
4   osmdata_sf()
5 base_ponto <- st_sf(q$osm_points)
```

Listing 4.1 – Base de dados de testes.

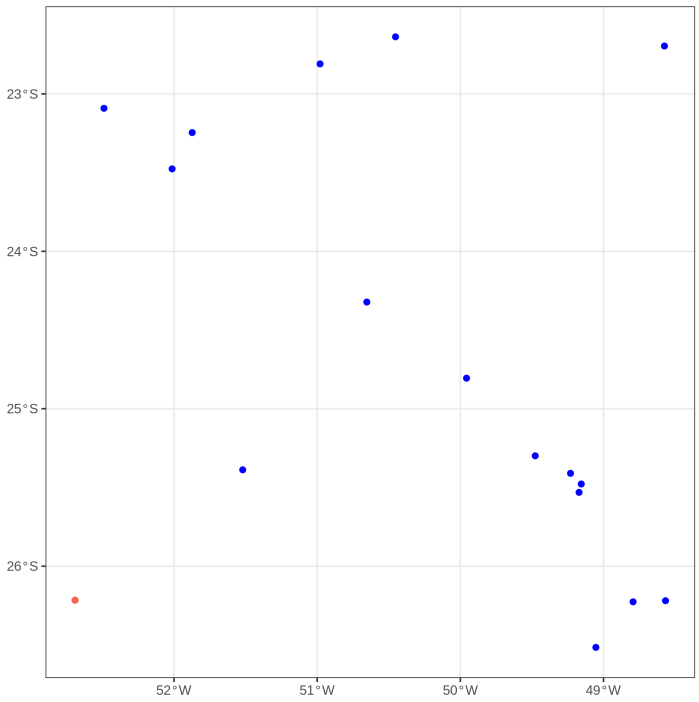
Esta base de dados contém 17 pontos, os quais podem ser observados na forma de tabela (Tabela 1) ou na forma de gráfico (Figura 3). Na Figura 3, como padronização, o ponto em vermelho será considerado o objeto de referências nas consultas deste capítulo. Ele é chamado 'search_obj', e é o primeiro dado geométrico da 'base_ponto' (Linha 5 do Listing 4.1). Os pontos azuis são os objetos geométricos restantes da 'base_ponto', e são chamados de 'data'. A dividão é feita de acordo com o Listing 4.2: search_obj é o ponto de referência do tipo *sfg*, e 'data' é a base de dados de vizinhança, e ela é do tipo *sf*.

Tabela 1 – Base de dados de teste em forma de tabela.

| | osm_id<br><chr> | aeroway<br><chr> | geometry<br><POINT [°]> |
|---|---|---|---|
| 1703178741 | 1703178741 | windsock | POINT (-52.69093 -26.21584) |
| 1827897936 | 1827897936 | windsock | POINT (-52.01297 -23.47645) |
| 1828488679 | 1828488679 | windsock | POINT (-51.87281 -23.24587) |
| 2036110871 | 2036110871 | windsock | POINT (-49.23037 -25.41025) |
| 5098201822 | 5098201822 | windsock | POINT (-50.97979 -22.80945) |
| 5176142434 | 5176142434 | windsock | POINT (-50.65306 -24.32234) |
| 5436268063 | 5436268063 | windsock | POINT (-48.79335 -26.22633) |
| 6385676288 | 6385676288 | windsock | POINT (-48.56694 -26.21941) |
| 6701982456 | 6701982456 | windsock | POINT (-51.51995 -25.38792) |
| 7299095704 | 7299095704 | windsock | POINT (-49.17082 -25.5311) |
| 8283556666 | 8283556666 | windsock | POINT (-48.57424 -22.69598) |
| 8305534029 | 8305534029 | windsock | POINT (-50.45275 -22.6376) |
| 8411187068 | 8411187068 | windsock | POINT (-49.47689 -25.29902) |
| 8773626181 | 8773626181 | windsock | POINT (-49.15557 -25.47792) |
| 8792963339 | 8792963339 | windsock | POINT (-49.0533 -26.51577) |
| 9277707008 | 9277707008 | windsock | POINT (-49.95672 -24.8057) |
| 9786255409 | 9786255409 | windsock | POINT (-52.48888 -23.09174) |

Fonte: A autora.

Figura 3 – Distribuição dos pontos da base de dados de teste




Fonte: autora.

```
search_obj <- base_ponto$geometry[[1]] #sfg
data <- base_ponto[2:17,] #sf
```

Listing 4.2 – Variáveis de teste

## 4.2 k-NN

A função k-NN faz uma busca dos vizinhos (pontos) mais próximos a um ponto de referência, a quantidade de vizinhos retornadas é k, onde k é um número inteiro maior ou igual a 1. Neste pacote, a consulta se chama sq_knn_query (Listing 4.3).

```
1  sq_knn_query <- function(data, search_obj, k){
2
3    if(!inherits(data, c("sf", "sfc"))){
4      stop("Parameter ‘data‘ must be either an sf or  sfc object.")
5    }
6    if(!inherits(search_obj, "sfg")){
7      stop("Parameter ‘search_obj‘ must be an sfg object.")
8    }
9    if(!is.wholenumber(k) && k >= 1){
10     stop("Parameter ‘k‘ have to be an integer greater than 1.")
11   }
12
13   #TODO improve the management of CRS of the search objects...
14   #should we consider that the search object has a CRS or not?
15   #or should it inherits from the data?
16   #or should we add an additional parameter for it?
17   search_obj_sfc <- st_sfc(list(search_obj), crs = st_crs(data))
18
19   dist <- st_distance(search_obj_sfc, data, by_element = TRUE)
20
21   if(nrow(st_sf(data)) < k ){
22     k <- nrow(data)
23   }
24
25   indices <- order(as.numeric(dist))[1:k]
26
27   if(inherits(data, "sf")){
28     data[indices, ]
29   }else{
30     data[indices]
31   }
32 }
```

Listing 4.3 – Consulta sq_knn_query

A consulta sq_knn_query recebe os seguintes parâmetros obrigatórios (linha 1):

- data - é a base de dados que contém todos os pontos que serão buscados;

- search_object - é o ponto de referência, são calculadas as distâncias de todos os pontos da 'data' com relação ao 'search_object';
- k - é a quantidade de pontos mais próximos que se quer obter.

O bloco entre as Linhas 3 e 11 são validações dos parâmetros de entrada: 'data' precisa ser do tipo *sf* ou *sfc*, 'search_obj' precisa ser do tipo *sfg* e k precisa sem um número inteiro maior do que zero. Caso algum desses requisitos não sejam cumpridos, é retornado ao usuário uma mensagem indicando que o valor da entrada não satisfaz a condição da consulta. A Linha 17 transforma o 'search_obj' num objeto *sfc* onde o CRS precisa ser igual ao do parâmetro 'data'. É importante o CRS do objeto de consulta e da base de dados de busca serem iguais. Na linha 19 são calculadas as distância entre o 'search_obj' com cada linha de 'data'. O bloco *if* que se inicia na Linha 21 valida se o valor de k excede a quantidade de itens que podem ser retornados. Caso 'data' possua menos pontos do que k, então k terá um novo valor, que é a quantidade de linhas que existem em 'data' A Linha 25 são ordena as distâncias obtidas na variável 'dist' (Linha 19) e armazena numa lista os indices das k distâncias mais próximas. No bloco *if/else* a partir da Linha 27, há uma verificação se 'data' é um *sf* ou um *sfc*, porque dependendo do seu tipo, a forma de retornar os valores é diferente. Porém, sendo outro o tipo da variável, são retornados para o resultado as linhas obtidas na variável 'indices'.

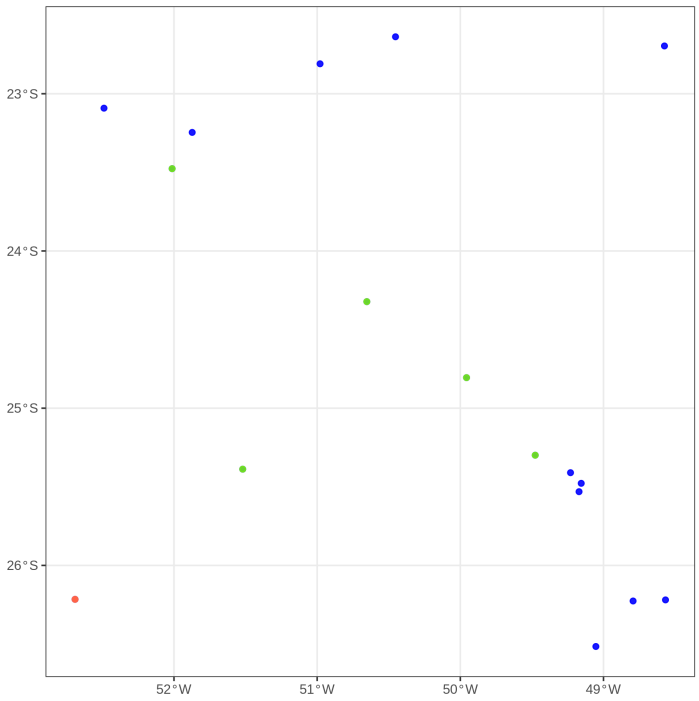
Para testar a consulta sq_knn_query, definiu-se k igual a 5, e o resultado obtido é o dataframe da Tabela 2. E a Figura 4 mostra visualmente os pontos retornados da consulta, eles são representados em verde, os pontos verdes e azuis pertencem a base de dados 'data' (estes pontos compõem a vizinhança, são os alvos), e o ponto vermelho é o ponto de referência (são buscados os k-NN deste ponto).

Tabela 2 – Resultado de sq_knn_query(data,search_obj, 5)

| | osm_id | aeroway | geometry |
|---|---|---|---|
| | <chr> | <chr> | <POINT [º]> |
| 6701982456 | 6701982456 | windsock | POINT (-51.51995 -25.38792) |
| 5176142434 | 5176142434 | windsock | POINT (-50.65306 -24.32234) |
| 1827897936 | 1827897936 | windsock | POINT (-52.01297 -23.47645) |
| 9277707008 | 9277707008 | windsock | POINT (-49.95672 -24.8057) |
| 8411187068 | 8411187068 | windsock | POINT (-49.47689 -25.29902) |

Fonte: A autora.

Figura 4 – Visualização da consulta sq_knn_query(data,search_obj, 5): pontos retornados da consulta são representados em verde, os pontos verdes e azuis compõem a vizinhança (são os alvos), e os ponto vermelho é o ponto de referência (são buscados os k-NN deste ponto).




Fonte: autora.

## 4.3 Range Query

A função Range Query faz uma busca dos vizinhos mais próximos a um ponto de referência, a quantidade de vizinhos retornadas é baseada numa distância 'dist'. Se um ponto A possui uma distância menor ou igual ao valor de 'dist' (com relação ao ponto de referência), então o ponto A faz parte do resultado. Neste trabalho, a consulta se chama sq_distance_query (Listing 4.4)

```
1 #' Implementation of distance-based query
2 #' @import sf
3 #' @export
4 sq_distance_query <- function(data, search_obj, dist){
5
6   if(!inherits(data, c("sf", "sfc"))){
7     stop("Parameter `data` must be either an sf or  sfc object.")
8   }
9   if(!inherits(search_obj, "sfg")){
10    stop("Parameter `search_obj` must be an sfg object.")
11  }
12  if(!is.numeric(dist) && dist >= 0){
13    stop("Parameter `dist` have to be a numeric greater than 0.")
14  }
```

```
15
16    search_obj_sfc <- st_sfc(list(search_obj), crs = st_crs(data))
17
18    dist2 <- st_distance(search_obj_sfc, data, by_element = TRUE)
19
20    sorted_objs <- sort(as.numeric(dist2), index.return = TRUE)
21
22    #select n items with distance less than dist
23    n <- 0
24    for(i in sorted_objs$x){
25      n <- n + 1
26      if(i > dist){
27        break
28      }
29     }
30
31     indices <- sorted_objs$ix[1:n-1]
32
33     if(inherits(data, "sfc")){
34       data[indices]
35     }else{
36       data[indices, ]
37     }
38 }
```

Listing 4.4 – Consulta sq_distance_query

A consulta sq_distance_query recebe os seguintes parâmetros obrigatórios (linha 4):

- data - é a base de dados que contém todos os pontos que serão buscados;
- search_object - é o ponto de referência, são calculadas as distâncias de todos os pontos da 'data' com relação ao 'search_object';
- dist - é a distância máxima que os pontos mais próximos precisam ter.

Esta consulta possui dependência da biblioteca sf (Linha 2). O bloco entre as Linhas 6 e 14 são validações dos parâmetros de entrada: 'data' precisa ser do tipo *sf* ou *sfc*, 'search_obj' precisa ser do tipo *sfg* e 'dist' precisa sem um número maior do que zero. Caso algum desses requisitos não sejam cumpridos, é retornado ao usuário uma mensagem indicando que o valor da entrada não satisfaz a condição da consulta. A Linha 16 transforma o 'search_obj' num objeto *sfc* onde o CRS precisa ser igual ao do parâmetro 'data'. É importante o CRS do objeto de consulta e base de dados de busca serem iguais. Na Linha 18 são calculadas as distância entre o 'search_obj' com cada linha de 'data'. A Linha 20 são ordena as distâncias obtidas na variável 'dist2' (Linha 18) e armazenas numa tabela com duas colunas (uma para as distância, e outra para os índices) Entre as Linhas 23 e 29, percorre-se a variável de distâncias ordenadas (sorted_objs), para cada linha, se a distância é menor ou igual a 'dist', é somado 1 a uma variável 'n' iniciada em zero. Quando a distância é maior que 'dist' o loop para. Na Linha 31 são pegos os primeiros indices obtidos na Linha 20, até o valor de 'n', obtido na Linha 25.

No bloco if/else a partir da Linha 33, há uma verificação se 'data' é um *sf* ou um *sfc*, porque dependendo do seu tipo a forma de retornar os valores é diferente. Porém, sendo um outro o tipo da variável, são retornados para o resultado as linhas obtidas na variável 'indices'.

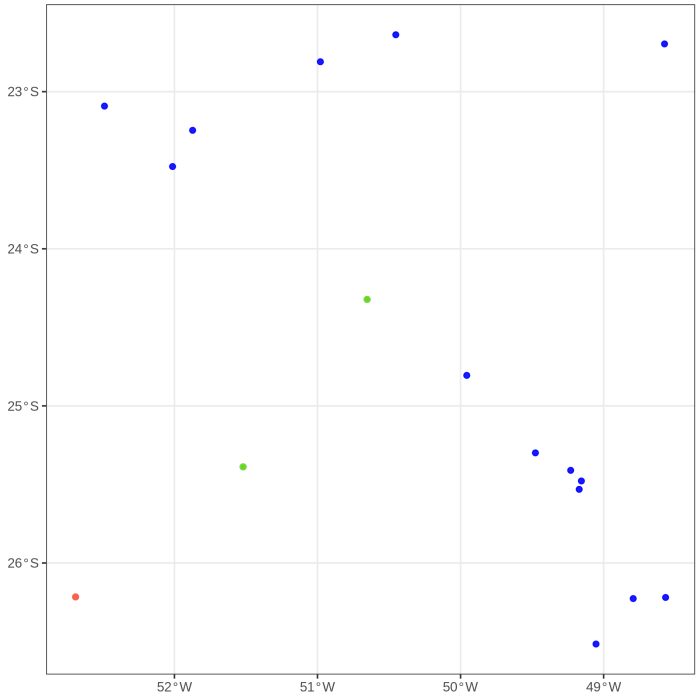
Para testar a consulta sq_distance_query, definiu-se 'dist' igual a 30.000,5 , e o resultado obtido é o dataframe da Tabela 3. E a Figura 5 mostra visualmente os pontos retornados da consulta, eles são representados em verde, os pontos verdes e azuis pertencem a base de dados 'data' (estes pontos compõem a vizinhança, são os alvos), e os ponto vermelho é o ponto de referência (a distância da Range Query é a partir deste ponto).

Tabela 3 – Resultado de sq_distance_query(data,search_obj, 30000.5)

| | osm_id | aeroway | geometry |
|---|---|---|---|
| | <chr> | <chr> | <POINT [°]> |
| 6701982456 | 6701982456 | windsock | POINT (-51.51995 -25.38792) |
| 5176142434 | 5176142434 | windsock | POINT (-50.65306 -24.32234) |

Fonte: A autora.

Figura 5 – Visualização da consulta sq_distance_query(data,search_obj, 30000.5): pontos retornados da consulta são representados em verde, os pontos verdes e azuis compõem a vizinhança (são os alvos), e o ponto vermelho é o ponto de referência (a distância da Range Query é a partir deste ponto).




Fonte: autora.

## 4.4 k-NN + Range Query

Esta consulta faz a busca dos vizinhos mais próximos a um ponto de referência combinando k-NN (Item 4.2) e Range query (Item 4.3). Neste trabalho, ela é chamada sq_range_knn_query (Listing 4.5)

```
1 #' Implementation of range kNN query
2 #' @import sf
3 #' @export
4 sq_range_knn_query <- function(data, search_obj, k, dist){
5   sq_distance_query(data, search_obj, dist) %>%
6     sq_knn_query(search_obj, k)
7 }
```

Listing 4.5 – Consulta sq_range_knn_query

Para esta consulta são necessários os parâmetros k (exclusivo da sq_knn_query), 'dist' (exclusivo da sq_range_query), 'data' e 'search_obj' (contidos das duas consultas)

A primeira etapa no código é executar a sq_distance_query(Linha 5) e o seu resultado é utilizado na sq_knn_query (Linha 6), o qual retorna o resultado da sq_range_knn_query.

Como característica, o resultado pode ser menor ou igual a k.

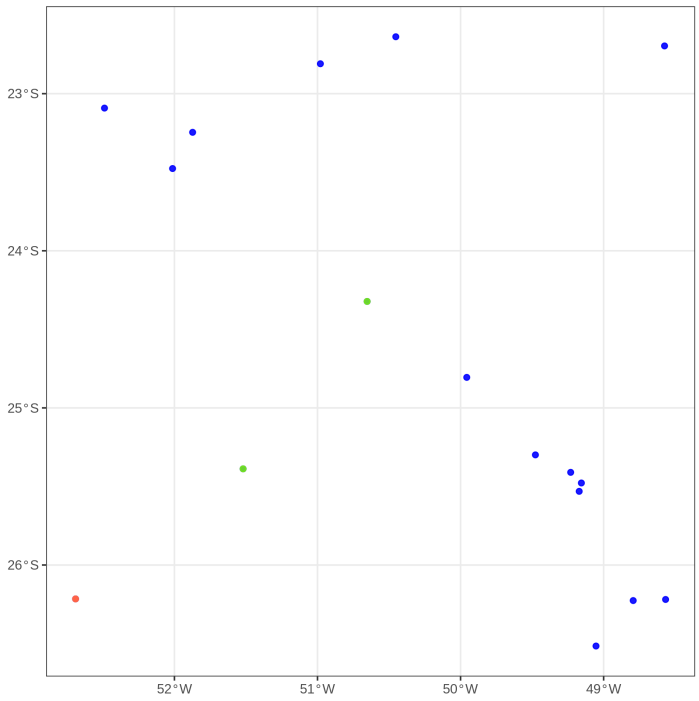
Para explificar a consulta sq_range_knn_query, são combinados os exemplos já apresentados da sq_distance_query e da sq_knn_query. Então k será 5 e dist será 30.000,5. O dataframe da consulta pode ser visto em Tabela 4. A Figura 6 mostra visualmente os pontos retornados da consulta, eles são representados em verde, os pontos verdes e azuis pertencem a base de dados 'data' (estes pontos compõem a vizinhança, são os alvos), e os ponto vermelho é o ponto de referência (o k-NN + Range Query é aplicado sobre este ponto). E como dito anteriormente, o resultado pode ser menor do k, mesmo que 'data' contenha mais linhas do que k. Porque 'data' pode ser reduzido pelo sq_distance_query.

Tabela 4 – sq_range_knn_query(data,search_obj, 5, 30000.5)

| | osm_id | aeroway | geometry |
|---|---|---|---|
| | <chr> | <chr> | <POINT [°]> |
| 6701982456 | 6701982456 | windsock | POINT (-51.51995 -25.38792) |
| 5176142434 | 5176142434 | windsock | POINT (-50.65306 -24.32234) |

Fonte: A autora.

Figura 6 – Visualização da consulta sq_range_knn_query(data,search_obj, 5, 30000.5): pontos retornados da consulta são representados em verde, os pontos verdes e azuis compõem a vizinhança (são os alvos), e o ponto vermelho é o ponto de referência (o k-NN + Range Query é aplicado sobre este ponto).




Fonte: autora.

## 4.5 Relative k-NN

A função Relative k-NN faz a busca dos vizinhos (pontos) mais próximos a um ponto da seguinte maneira:

- pega-se a distância do k item mais próximo ao ponto de referência;
- sobre esta distância é acrescida uma porcentagem, gerando uma nova distância;
- então, é retornado todos os pontos da base de busca com distância menor ou igual à nova distância.

Neste trabalho, esta consulta é chamada sq_relative_knn_query (Listing 4.6)

```
1 #' Implementation of relative kNN query
2 #' @import sf
3 #' @export
4 sq_relative_knn_query <- function(data, search_obj, k, dist_buff){
5
6   if(!is.numeric(dist_buff) && dist_buff >= 0){
7      stop("Parameter 'dist_buff' have to be a numeric greater than 0.")
8    }
9
10  result_knn <- sq_knn_query(data, search_obj, k)
11
```

```
12   if(inherits(result_knn, "sf")){
13     geometries <- st_geometry(result_knn)
14   } else {
15     geometries <- result_knn
16   }
17
18   search_obj_sfc <- st_sfc(list(search_obj), crs = st_crs(data))
19   obj_aux <- st_sfc(list(geometries[[length(geometries)]]),
20                     crs = st_crs(data))
21
22   max_dist <- st_distance(search_obj_sfc, obj_aux, by_element = TRUE)
23   max_dist_buff <- as.numeric(max_dist[[1]] * (1 + dist_buff/100))
24
25   sq_distance_query(data, search_obj, max_dist_buff)
26 }
```

Listing 4.6 – Consulta sq_relative_knn_query

Esta consulta possui dependência da biblioteca sf (Linha 2) e das consultas sq_knn_query (Linha 10 - ver o Item 4.2) e sq_distance_query (Linha 24 - ver o Item 4.3).

Os parâmetros obrigatórios (Linha 4) são:

- data - é a base de dados que contém todos os pontos que serão buscados;
- search_object - é o ponto de referência, são calculadas as distâncias de todos os pontos da 'data' com relação ao 'search_object';
- k - é a quantidade de pontos mais próximos que se quer obter (ver item 4.2);
- dist_buff - porcentagem de acréscimo sobre a maior distância obtida.

No bloco da Linha 6 é feita uma validação sobre o valor da variável dist_buff, cuja precisa ter um valor maior que zero. Na linha 10 é chamada a função sq_knn_query, onde são validados os outros parâmetros. Caso algum deles não atendam aos requisitos, é retornado ao usuário uma mensagem informando o erro. A utilização da sq_knn_query na linha 10 é para que seja obtido k vizinho mais próximo (ou ponto mais distante, de acordo com k). O resultado é a variável result_knn. O bloco if/else da Linha 12 valida de a saída result_knn é um *sf* ou *sfc*, porque é necessário termos acesso apenas à coluna geométrica. A coluna geométrica é armazenada na variável 'geometries Na Linha 18 o 'search_object' é transformado em um *sfc*, esta transformação é necessária para a Linha 21, ao se calcular a distância. A Linha 19 faz o processo igual ao da Linha 18, porém para o último ponto do resultado da sq_knn_query. Na Linha 21, calcula-se a distancia entre o ponto k mais distante (obj_aux) do ponto de referência (search_obj_sfc). E na Linha 22 é acrescido a porcentagem 'dist_buff' sobre esta distância obtida, gerando a variável max_dist_buff. Depois é executada a consulta sq_distance_query, com a nova distância (max_dist_buff). Nesta consulta o resultado pode ser maior do k. Caso k seja menor que a quantidade de pontos existentes em 'data'.

Para exemplificar a consulta sq_relative_knn_query, k é 2 e 'dist_buff' é de 10%. O resultado da consuta pode ser visto na Tabela 5 e a Figura 7 mostra visualmente os pontos

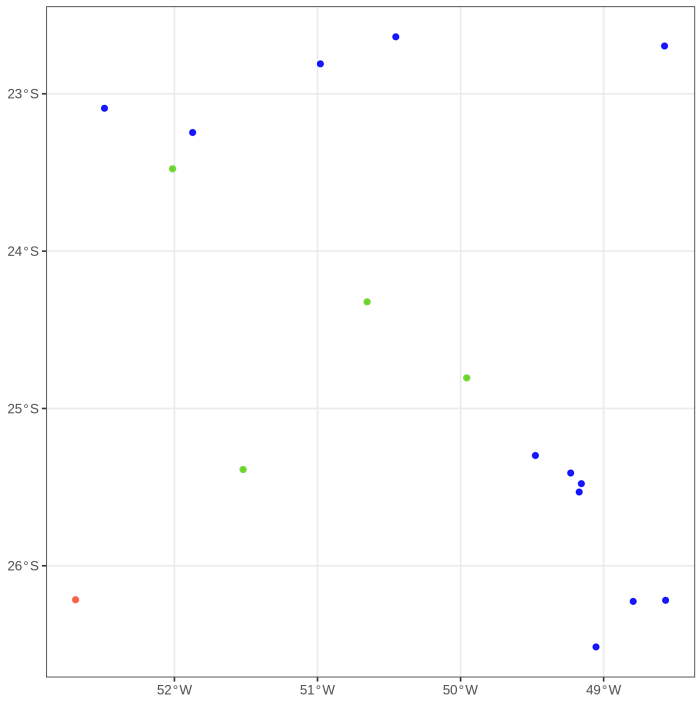
retornados da consulta, eles são representados em verde, os pontos verdes e azuis pertencem a base de dados 'data' (estes pontos compõem a vizinhança, são os alvos), e os ponto vermelho é o ponto de referência (o Relative k-NN é aplicado sobre este ponto).

Tabela 5 – Resultado de sq_relative_knn_query(data,search_obj, 2, 10)

| | osm_id | aeroway | geometry |
|---|---|---|---|
| | <chr> | <chr> | <POINT [°]> |
| 6701982456 | 6701982456 | windsock | POINT (-51.51995 -25.38792) |
| 5176142434 | 5176142434 | windsock | POINT (-50.65306 -24.32234) |
| 1827897936 | 1827897936 | windsock | POINT (-52.01297 -23.47645) |
| 9277707008 | 9277707008 | windsock | POINT (-49.95672 -24.8057) |

Fonte: A autora.

Figura 7 – Visualização da consulta sq_relative_knn_query(data,search_obj, 2, 10): pontos retornados da consulta são representados em verde, os pontos verdes e azuis compõem a vizinhança (são os alvos), e o ponto vermelho é o ponto de referência (o Relative k-NN é aplicado sobre este ponto).




Fonte: autora.

## 4.6 Reverse k-NN

Na função Reverse k-NN, para cada ponto da base alvo, é calculado o seu k-NN. E se o ponto de referência estiver contido no resultado do k-NN do ponto alvo, então o

ponto alvo faz parte do resultado do Reverse k-NN. Neste trabalho, esta consulta se chama sq_reverse_knn_query (Listing 4.7)

```
#' Implementation of reverse kNN query
#' @import sf
#' @export
sq_reverse_knn_query<- function(data, search_obj, k){

  if(!inherits(data, c("sf", "sfc"))){
    stop("Argumento 'data' precisa ser do tipo sf.")
  }
  if(!inherits(search_obj, "sfg")){
    stop("Argumento 'search_obj' precisa ser do tipo sfg.")
  }
  if(!is.wholenumber(k)){
    stop("Argumento 'k' precisa ser do tipo int.")
  }

  isIn <- FALSE
  indexes <- c()

  if(inherits(data, "sf")){
    geometries <- st_geometry(data)
  }else{
    geometries <- data
  }
  len <- length(geometries)

  search_obj_sfc <- st_sfc(list(search_obj), crs = st_crs(data))

  if(any(geometries==search_obj_sfc)){
    isIn <- TRUE
  }

  if(isIn == FALSE){
    geometries <- st_sfc(c(geometries, search_obj_sfc))
  }

  for(i in 1:len){
    result_knn <- sq_knn_query(geometries, geometries[[i]], (k+1))

    for(j in 1:length(result_knn)){
      if(result_knn[[j]] == search_obj){
        indexes <- append(indexes, i)
      }
    }
  }

```

```
46   if(inherits(data, "sfc")){
47       data[indexes]
48   }else{
49       data[indexes, ]
50   }
51 }
```

Listing 4.7 – Consulta sq_reverse_knn_query

Esta consulta possui depêndencia da biblioteca sf (Linha 2) e da consulta sq_knn_query (Linha 39 - ver o Item 4.2).

A consulta sq_reverse_knn_query recebe os seguintes parâmetros obrigatórios (Linha 4):

- data - é a base de dados que contém todos os pontos que serão buscados;
- search_object - é o ponto de referência, são calculadas as distâncias de todos os pontos da 'data' com relação ao 'search_object';
- k - é a quantidade de pontos mais próximos que se quer obter.

O bloco entre as Linhas 6 e 14 são validações dos parâmetros de entrada: 'data' precisa ser do tipo sf ou sfc, 'search_obj' precisa ser do tipo *sfg* e k precisa sem um número inteiro maior do que zero. Caso algum desses requisitos não sejam cumpridos, é retornado ao usuário uma mensagem indicando que o valor da entrada não satisfaz a condição da consulta. Após a validação dos parâmetros é feita uma verificação para saber se 'search_obj' está dentro de 'data'. Para isso, primeiro é obtida a coluna geométrica de 'data' (bloco *if/else* da Linha 19). Depois, na Linha 26, tranforma-se 'search_obj' em sfc, mesma classe da coluna geométrica obtida anteriormente. Então, na Linha 28, é verificado se o ponto alvo está contido na base alvo. Caso não esteja, o ponto alvo é inserido no fim do *sfc* de pontos alvos (Linha 33). Agora é feita a consulta sq_knn_query para todos os pontos da base alvo (bloco da Linha 36). Porém, caso 'search_obj' tenha sido inserido dentro desta consulta (sq_reverse_knn_query), não é feito o k-NN dele. Porque este ponto não pertence de fato a esta base alvo. Dentro do bloco da Linha 36, para cada kNN feito (Linha 37) é verificado se no resultado está contido o ponto de referência. Caso esteja, é guardado o índice do ponto alvo na variável 'indexes' (Linha 41). Por fim, é retornado o resultado em formato *sf* ou *sfc*, bloco *if/else* da Linha 46.

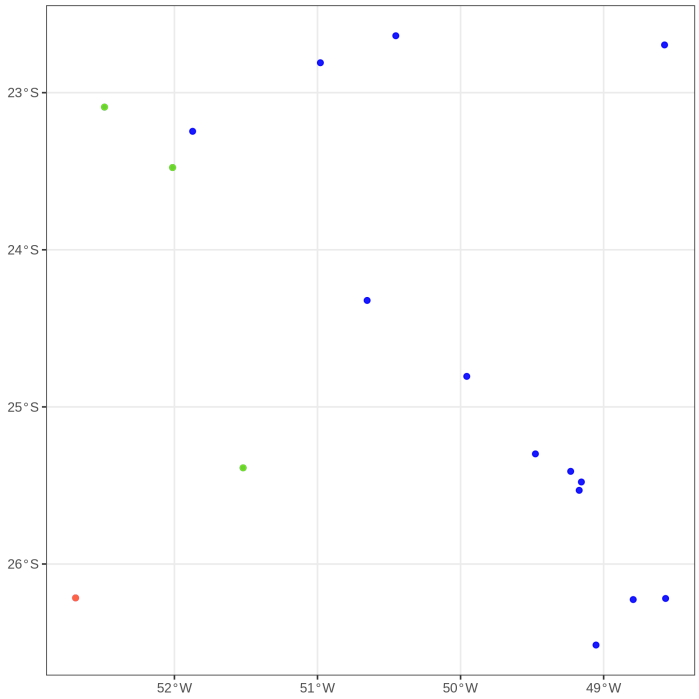
Para exemplificar a consulta sq_reverse_knn_query, k será igual a 8. A Tabela 6 mostra a saída da chamada da consulta, e a Figura 8 é a parte visual de como ficou a distribuição dos pontos: os pontos retornados da consulta, eles são representados em verde, os pontos verdes e azuis pertencem a base de dados 'data' (estes pontos compõem a vizinhança, são os alvos), e os ponto vermelho é o ponto de referência (o Reverse k-NN é aplicado sobre este ponto).

Tabela 6 – Resultado de sq_reverse_knn_query(data,search_obj, 8)

| | osm_id <chr> | aeroway <chr> | geometry <POINT [°]> |
|---|---|---|---|
| 1827897936 | 1827897936 | windsock | POINT (-52.01297 -23.47645) |
| 6701982456 | 6701982456 | windsock | POINT (-51.51995 -25.38792) |
| 9786255409 | 9786255409 | windsock | POINT (-52.48888 -23.09174) |

Fonte: A autora.

Figura 8 – Visualização da consulta sq_reverse_knn_query(data,search_obj, 8): pontos retornados da consulta são representados em verde, os pontos verdes e azuis compõem a vizinhança (são os alvos), e o ponto vermelho é o ponto de referência (o Reverse k-NN é aplicado sobre este ponto).




Fonte: autora.

## 4.7 k Closest Pairs

Dado duas bases de pontos (base A e base B), são calculadas as distâncias entre cada ponto da base A com cada ponto da base B. Na função k Closest Pairs, sendo k um número inteiro maior do que zero, é retornado os k pares mais próximos, de acordo com as distâncias calculadas. Neste trabalho esta consulta é chamada sq_k_closest_pairs_query (Listing 4.8)

```
1 #' Implementation of k closest pairs query
2 #' @import sf
3 #' @export
4 sq_k_closest_pairs_query <- function(data_ref, data_search, k){
```

```
5
6   if(!inherits(data_ref, c("sf", "sfc"))){
7     stop("Argumento 'data_ref' precisa ser do tipo sf.")
8   }
9   if(!inherits(data_search, c("sf", "sfc"))){
10    stop("Argumento 'data_search' precisa ser do tipo sf.")
11  }
12  if(!(st_crs(data_ref) == st_crs(data_search))){
13    stop(" O CRS entre 'data_ref' e 'data_search' precisa ser igual.")
14  }
15
16  df <- data.frame(matrix(ncol = 3, nrow = 0))
17  colnames(df) <-c("ref", "target", "dist")
18
19  if(inherits(data_search, "sf")){
20    geometries_ref <- st_geometry(data_ref)
21    geometries <- st_geometry(data_search)
22  }else{
23    geometries_ref <- data_ref
24    geometries <- data_search
25  }
26
27  len_ref <- length(geometries_ref)
28
29  for(i in 1:len_ref){
30    search_obj_sfc <- st_sfc(list(geometries_ref[[i]]), crs = st_crs(
     data_ref))
31    dist <- st_distance(search_obj_sfc, data_search, by_element = TRUE,)
32    sorted_objs <- sort(as.numeric(dist), index.return = TRUE)
33
34    for(j in 1:length(sorted_objs$x)){
35      df[nrow(df) + 1,] <- c(list(geometries_ref[i]), list(geometries[
     sorted_objs$ix[j]]), sorted_objs$x[j])
36    }
37  }
38
39  indexes <- order(as.numeric(df$dist))[1:k]
40
41  result <- df[indexes,]
42  result$ref <- st_sfc(result$ref)
43  result$target <- st_sfc(result$target)
44  result$dist <- NULL
45  rownames(result) <- NULL
46  result
47 }
```

Listing 4.8 – Consulta sq_k_closest_pairs_query

Esta consulta tem dependência da biblioteca *sf* (Linha 2). E os seus parâmetros obrigatórios (Linha 4) são:

- data_ref - é a base de dados que contém todos os pontos referência;
- data_search - é a base de dados que contém todos os pontos alvo;
- k - é a quantidade de pares de pontos mais próximos que se quer obter.

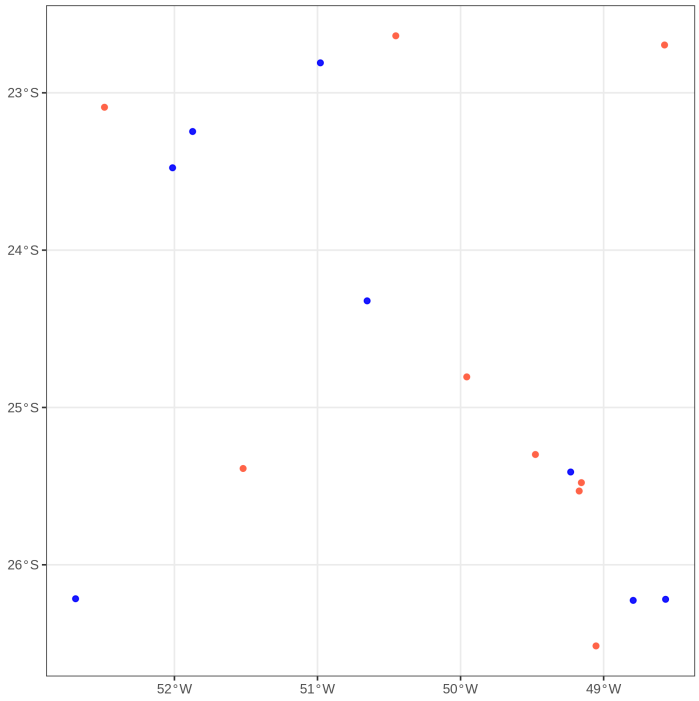
Entre as Linhas 6 e 14 sao feitas as validações dos parâmetros de entrada. As variáveis 'data_ref' e 'data_search' precisam ser do tipo *sf* ou *sfc* e o CRS delas precisa ser o mesmo. O bloco *if/else* da Linha 19 verifica se 'data_ref' e 'data_search' são *sf* ou *sfc* para pegar apenas a coluna geométrica dessas variáveis. Na Linha 27 define-se a variável 'len_ref' para armazenar a quantidade de pontos existentes na base de referência No bloco da linha 29, cada ponto de referência é transformado em um *sfc* (Linha 30), depois são calculadas as distâncias deste ponto com todos os outros da base alvo (Linha 31). Depois, estes pontos são ordenados (Linha 32), e inseridos dentro de um dataframe 'df' (bloco da Linha 34, o dataframe aqui utilizado foi definido na Linha 16). Este dataframe contém 3 colunas: ref, target e dist, onde é armazena, respectivamente, o índice do ponto de referência, o índice do ponto alvo, e a distância entres estes dois pontos. Na Linha 39, todas as combinações entre os pontos de 'data_ref' e 'data_search' são ordenados pela distância, e armazena-se apenas os k índices de 'df' na variável 'indexes' Na linha 41 são recuperados todos os valores de 'df' que correspondem aos valores de 'indexes'. Na Linha 42 e 43 as colunas ref e target recuperados são transformadas em *sfc*. Remove-se a coluna dist (Linha 44) e os índices (Linha 45) do resultado.

Para exemplificar a consulta sq_k_closest_pairs_query os pontos obtidos em Listing 4.1 foram divididos em dois (Listing 4.9). E os dois grupos ficaram como na Figura 9

```
1 data_ref <- base_ponto[1:8,]
2 data_search <- base_ponto[9:17,]
```

Listing 4.9 – Base de dados para sq_k_closest_pairs_query

Figura 9 – Divisão da base em dois grupos




Fonte: autora.

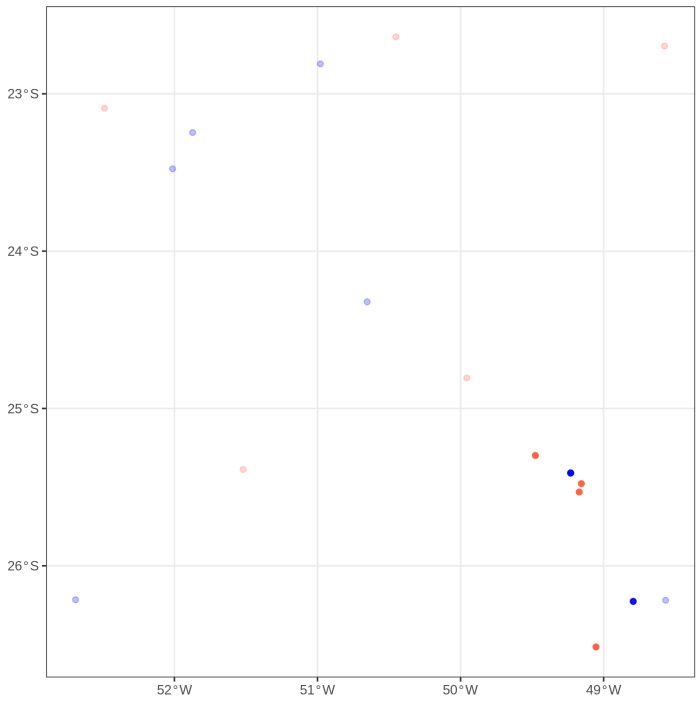
Chamando a consulta com k igual a 4, a saída obtida para sq_k_closest_pairs_query(data_ref, data_search, 4) pode ser vista em Tabela 7. E a vizualização destes pontos pode ser vista na Figura 10, na qual os pontos em azul compõem a base de dados de referência (sobre cada ponto é calculado o k-NN), os pontos em vermelho são a base de dados de alvo ou vizinhança, e os pontos com cor sólida fazem parte do resultado da consulta.

Tabela 7 – Consulta sq_k_closest_pairs_query(data_ref, data_search, 4)

| ref <POINT> | target <POINT> |
|---|---|
| POINT (-49.27532 -25.43777) | POINT (-49.15557 -25.47792) |
| POINT (-49.27532 -25.43777) | POINT (-49.17082 -25.5311) |
| POINT (-49.27532 -25.43777) | POINT (-49.47689 -25.29902) |
| POINT (-48.79335 -26.22633) | POINT POINT (-49.0533 -26.51577) |

Fonte: A autora.

Figura 10 – Visualização da consulta sq_k_closest_pairs_query(data_ref, data_search, 4): os pontos em azul compõem a base de dados de referência (sobre cada ponto é calculado o k-NN), os pontos em vermelho são a base de dados de alvo ou vizinhança, e os pontos com cor sólida fazem parte do resultado da consulta.




Fonte: autora.

# 5 CONCLUSÃO

Neste trabalho são apresentados os algoritmos de consultas espaciais baseadas em métrica (onde a distância entre um ponto e outro é relevante) utilizando a linguagem R. Para o desenvolvimento, foi utilizado o tipo de dado geométrico ponto, porém não é definitivo. Os algoritmos deste trabalho aceitam também os outros tipos de dados geométricos (por exemplo, linha), mas pode ocorrer um resultado insatisfatório, pois durante a elaboração das consultas não se levou em consideração a extensão dos objetos. Os testes apresentados foram sobre bases de dados não volumosas, para que a visualização da aplicação de cada algoritmo ficasse mais simples. Então, é interessante num projeto futuro testes em bases volumosas, para se observar o desempenho destas consultas, e se necessário estudar ajustes para melhorar a performance. Além desta sugestão, pode-se incluir para oportunidades futuras:

- a inclusão das consultas desenvolvidas neste trabalho em um pacote da linguagem R;
- os algoritmos foram desenvolvidos levando em consideração pontos. Como forma de expandir estas consultas para os outros tipos de dados espaciais vetorias, pode-se levar em consideração alguns ajustes para que as consultas envolvendo linhas, por exemplo, retornem resultados mais adequados;
- inclusão de estruturas de indexação, para proporcionar aceleração no processamento

# 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**. New York, NY, USA: Association for Computing Machinery, 2020. (MEDES '20), p. 10–17. ISBN 9781450381154. Disponível em: <https://doi.org/10.1145/3415958.3433038>. Citado 2 vezes nas páginas 9 e 10.

CASANOVA, M. A. et al. **Bancos de Dados Geográficos**. Ed. Mundo Geo, 2005. Disponível em: <http://www.dpi.inpe.br/livros/bdados/cap8.pdf>. Acesso em: 08 de outubro de 2022. Citado na página 11.

CORRAL, A. et al. Closest pair queries in spatial databases. In: **Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data**. New York, NY, USA: Association for Computing Machinery, 2000. (SIGMOD '00), p. 189–200. ISBN 1581132174. Disponível em: <https://doi.org/10.1145/342009.335414>. Citado na página 13.

CÂMARA, G. et al. **Anatomia de Sistemas de Informação Geográfica**. UNICAMP Editora, 1996. Disponível em: <http://www.dpi.inpe.br/gilberto/livro/anatomia.pdf/>. Acesso em: 08 de outubro de 2022. Citado na página 9.

DAVIS JR, C. A. et al. **O Open Geospatial Consortium**. [s.n.], 2005. Disponível em: <http://www.dpi.inpe.br/livros/bdados/cap11.pdf>. Acesso em: 08 de outubro de 2022. Citado na página 11.

GÜTING, R. H. An introduction to spatial database systems. **VLDB**, v. 4, n. 3, 1994. Citado 2 vezes nas páginas 7 e 9.

MARINO, T. B. **Representação de Dados Espaciais - Raster x Vetor x TIN**. 2000. Disponível em: <http://www.ufrrj.br/lga/tiagomarino/aulas/5%20-%20Representacao%20de%20Dados%20Espaciais%20-%20Raster%20x%20Vetor%20x%20TIN.pdf>. Acesso em: 08 de outubro de 2022. Citado na página 7.

PEBESMA, E. **Simple Features for R**. 2022. Disponível em: <https://cran.r-project.org/web/packages/sf/vignettes/sf1.html>. Acesso em: 08 de outubro de 2022. Citado 3 vezes nas páginas 7, 12 e 13.

POSTGIS. **PostGIS**. 2022. Disponível em: <https://postgis.net/>. Acesso em: 08 de outubro de 2022. Citado na página 12.

TANIAR, D.; RAHAYU, W. A taxonomy for nearest neighbour queries in spatial databases. **Journal of Computer and System Sciences**, v. 79, n. 7, p. 1017–1039, 2013. ISSN 0022-0000. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0022000013000275>. Citado na página 13.