Interpretabilidade de Dados em Portais de Dados Abertos Georreferenciados, Um Estudo de Caso

Ítalo L. F. Portinho[1], Lucas Natan[1], Ruan Pablo[1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
Av. Gal. Milton Tavares de Souza, s/nº – 24.210-346 – Niterói – RJ – Brazil

italons@gmail.com, {italoleite, lucasnatan, ruanpablo}@id.uff.br

Abstract. *Starting from the definition of data interpretability established in previous works, and a framework of desirable features, this article analyzes the ecosystem of applications within the Infraestrutura Nacional de Dados Espaciais, known as INDE, with a focus on the application called Visualizador, a portal for georeferenced open data. As a case study, we began with a dataset of crimes in the city of São Paulo, imported into a data warehouse for processing large volumes of data queries in parallel and generating new datasets. The geometry was converted using Quantum GIS software, merged with a layer containing the geometries of São Paulo's districts, and exported in GeoJSON format for visualization on a map. This process demonstrates the importance of concepts such as open data, transparency, data quality, and interoperability among portals for open data and their various data formats in producing valuable knowledge for the population.*

Resumo. *Partindo de definição de interpretabilidade de dados definida em trabalhos anteriores, e um framework de características desejáveis, este artigo analisa o ecossistema de aplicações da Infraestrutura Nacional de Dados Espaciais, a INDE, e em especial a aplicação chamada Visualizador, um portal de dados abertos georreferenciados. Como estudo de caso partimos de um dataset de crimes da cidade de São Paulo, importado em um datawarehouse para processar consultas de grandes volumes de dados em paralelo e gerar novos datasets, com sua geometria convertida no software Quantum GIS, e mesclados com uma camada com a geometria dos distritos de São Paulo, e exportados no formato GeoJSON, para podermos visualizar em um mapa a informação produzida. Este processo demostra a importância dos conceitos de dados abertos, transparência, boa qualidade do dado, e interoperabilidade entre portais de dados abertos e seus diversos formatos de dados, na produção de conhecimento útil para a população.*

1. Introdução

O acesso aberto aos dados governamentais traz benefícios significativos para a sociedade, permitindo tanto a fiscalização cidadã quanto uma análise aprofundada dos problemas sociais. No entanto, a interpretabilidade desses dados é crucial para que seu potencial seja plenamente aproveitado. Embora seja reconhecida como um elemento importante da qualidade dos dados, ainda não há um consenso sobre sua definição conceitual. O trabalho apresentado em [1] visa definir interpretabilidade de dados.

Dois temas importantes para embasar o trabalho são a transparência de dados e a interação humano-dado. O artigo [1] identificou algumas características importantes da transparência de dados como disponibilidade sem restrições, confiabilidade, fácil acesso e entendimento da informação, objetividade e canais de comunicação abertos. A transparência, entretanto, depende da capacidade dos usuários de interpretar os dados de forma eficiente, o que ressalta a necessidade de uma definição formal de interpretabilidade de dados. Além disso, a interação humano-dado (IHD) é uma área de pesquisa que estuda a manipulação e análise de dados complexos, destacando a legibilidade e a agência como aspectos fundamentais para atingir os objetivos de transparência. Para identificar as diferentes definições de interpretabilidade de dados, foi realizada uma revisão de literatura utilizando critérios específicos de busca, resultando em 32 trabalhos analisados. As definições de interpretabilidade foram então organizadas e relacionadas a um framework de requisitos não-funcionais (NFR Framework), agrupando-as em contextos relevantes para os portais de dados abertos.

Questionários foram aplicados a estudantes de pós-graduação e professores com experiência em dados governamentais abertos para complementar e priorizar as características encontradas. Foi realizada também uma pesquisa envolvendo diversos especialistas na área de análise de dados a fim de complementar e ordenar as ordens dos grupos de características.

Foi proposto por [1] 8 grandes grupos de características desejáveis:

- G1 Compreensibilidade, simplicidade, claridade e redigibilidade;
- G2 Confiabilidade e rastreabilidade;
- G3 Estrutura e Organização;
- G4 Acurácia e corretude;
- G5 Completude;
- G6 Concisão;
- G7 Consistência e coerência ;
- G8 Informatividade

A definição encontrada para a Interpretabilidade de dados foi: “Interpretabilidade é a capacidade de um conjunto de dados, com total precisão, consistência, coerência e organização, de transmitir significado de forma clara e compreensível ao usuário” [1]. Um estudo foi realizado em portais de dados abertos do Brasil, Europa e Estados Unidos. A conclusão deste estudo revelou que há um grande volume de dados acumulados com pouco valor nesses portais, principalmente devido à publicação de dados que não estão adequados para serem disponibilizados.

2. A Infraestrutura Nacional de Dados Espaciais(INDE)

A Infraestrutura Nacional de Dados Espaciais – INDE foi instituída pelo Decreto Nº 6.666 de 27/11/2008 com a seguinte definição: Conjunto integrado de tecnologias; políticas; mecanismos e procedimentos de coordenação e monitoramento; padrões e acordos, necessário para facilitar e ordenar a geração, o armazenamento, o acesso, o compartilhamento, a disseminação e o uso dos dados geoespaciais de origem federal, estadual, distrital e municipal. A INDE foi concebida com o propósito de catalogar, integrar e harmonizar dados geoespaciais produzidos ou mantidos e geridos nas instituições de governo brasileiras, de modo que possam ser facilmente localizados, explorados em suas características e acessados para os mais variados fins por qualquer usuário com acesso à Internet. A catalogação dos dados geoespaciais é feita mediante seus respectivos metadados pelos próprios produtores e/ou gestores dos dados [2].

O Ecossistema de aplicações da INDE abertas ao público é composto pelo Portal, Visualizador, Catálogo de Metadados e Catálogo de Geosserviços. Este trabalho visa analisar a interpretabilidade dos dados apenas do Visualizador.

2.1. Portal de INDE:

É um site que agrega todas as funcionalidades da INDE, links para todas as aplicações, eventos da INDE, contato, suporte, notícias e uma seção chamada área de download. Essa seção lista as camadas que as instituições aderentes cadastraram em seus metadados, e um backend responsável pela integração lê essas camadas e faz a associação metadado x camada em um banco de dados. A área de download também tem um *preview* de cada camada, e disponibiliza o download dela em diversos formatos.

2.2. Visualizador:

O Visualizador da INDE oferece aos produtores e usuários de informações geoespaciais mecanismos que permitam explorar os catálogos de metadados e geosserviços localizados em servidores pertencentes a diferentes organizações e instituições. A aplicação possibilita a criação de mapas e a execução de diferentes funções de visualização e navegação. Abaixo algumas de suas funcionalidades [5]:

- Consulta por Instituição e Tema;
- Mecanismo que permita combinar diversos temas para a construção de uma consulta;
- Mecanismos para impressão dos resultados das consultas;
- Mecanismos que permite realizar consultas com filtros por atributos, sua exportação em formato csv e a espacialização do resultado da consulta ;
- Exportação das camadas em formatos csv, kml e shp;
- Mecanismos de comunicação com outras IDEs e plataformas abertas (OSM);
- Exibe/esconde as toponímias;
- Mecanismo de visualização de camadas sobrepostas por deslizamento da janela da camada mais ao topo e transparência;
- Mecanismo de busca de localidades e aproximação para a localidade de interesse.

2.3. Catálogo de Metadados:

Na prática, os metadados visam descrever, localizar, facilitar a recuperação e gerência de um recurso de informação. Assim, para que os metadados, escritos segundo o perfil de Metadados Geoespaciais do Brasil (MGB), possam, efetivamente, alcançar esses propósitos, o perfil MGB foi implantado em um software chamado Geonetwork. O GeoNetwork é um catálogo de metadados livre, de código aberto, distribuído, inicialmente, pela FAO/ONU [3]. Essas características (livre e de código aberto) permitiram que o mesmo fosse customizado para atender as necessidades brasileiras.

Por estar aderente aos padrões adotados na INDE e por ser um software de livre distribuição, o GeoNetwork é a ferramenta recomendada no plano de ação para a implantação da INDE para carga e gestão de metadados geoespaciais. Entre as principais características do catálogo estão [3]:

- A utilização de protocolos e ferramentas que permitem a implantação de uma rede distribuída de metadados entre diferentes nós participantes de uma rede;
- A implementação de níveis de segurança permitindo a definição de grupos e papéis e seus privilégios para a edição, consulta e disseminação de metadados;
- Uma interface globalizada, que permite o acesso aos metadados nos idiomas português-br, inglês e espanhol;
- A recuperação dos metadados através de mecanismos de busca avançada, que permitem a busca por elementos como as categorias de informação (ex: Solos, Altimetria, Vegetação, etc), retângulo envolvente do produto documentado, palavra-chave, etc.;
- A carga e exibição de metadados nos principais padrões internacionais: ISO-19115/ 19139, FGDC e Dublin-Core;
- A adesão a padrões de serviços OGC (Open Geospatial Consortium).

2.4. Catálogo de Geosserviços:

Geosserviços são serviços web específicos para o domínio geoespacial, constituindo um poderoso conjunto de funcionalidades para coletar, armazenar, recuperar sem restrições, transformar e apresentar dados espaciais associados a um determinado objetivo. Essas funcionalidades são usadas por meio de um navegador Web ou outra aplicação qualquer (QGIS ou um aplicativo em Smartphone, por exemplo). Através dos geosserviços é possível exportar informações geoespaciais para uma ampla gama de formatos, tanto para edição de objetos como para a apresentação de mapas. A INDE adotou os padrões do *Open Geospational Consortium* OGC para a especificação de seus geosserviços [4]. Abaixo são listadas as implementações disponíveis desses padrões:

- WMS (*Web Map Service*): define um geosserviço para uma representação visual dos dados espaciais em algum formato de imagem e não os dados em si. Estas representações serão geradas no formato de imagem, como JPEG, PNG e GIF ou em formato vetorial, como o *Scalable Vector Graphics* (SVG). Este padrão especifica como o cliente deve requisitar as informações para o servidor e como este deve responder ao cliente. As operações WMS podem ser realizadas a partir de um navegador web que fará a submissão das requisições sob a forma de uma URL.

- WFS (*Web Feature Service*): define um serviço para que clientes possam recuperar feições espaciais em formato GML. As operações WFS podem ser realizadas a partir de um navegador web que fará a submissão das requisições sob a forma de uma URL e este retornará os dados em si.
- WCS (*Web Covarage Service*): define o acesso aos dados que representam fenômenos com variação contínua no espaço. Este serviço é especificado para tratamento de dados modelados como geocampos.

2.5. Interpretabilidade de dados abertos no Visualizador da INDE:

O Visualizador é um agregador dos dados produzidos pelas instituições aderentes a INDE. Vamos listar abaixo como o visualizador se encaixa em cada característica do modelo de interpretabilidade de dados proposto pelo artigo base.

- Compreensibilidade, simplicidade, clareza e legibilidade: sendo um portal de dados especializados, espera-se que o usuário seja iniciado em GIS. No entanto, o site é de fácil uso. Os conceitos associados aos dados da camada devem ser acessados nos seus metadados, porém o visualizador não oferece tradução para outros idiomas;
- Confiabilidade e rastreabilidade: uma vez que todos os dados do visualizador são providos pelas instituições que os produziram, a fonte do dado é bem listada. Porém a INDE não opera com captação de dados de proveniência em banco de dados;
- Estruturação e organização: o visualizador possui um menu com as camadas divididas por temas, no entanto essa forma de acesso parece depreciada em relação à outra forma de acesso mais usada, a leitura do catálogo de dados das instituições por uma requisição GetFeature, que lista todas as camadas. Na forma de acesso por temas, diversas camadas não funcionam mais e estão presentes instituições que não pertencem mais a INDE.
- Precisão e correção: não está presente;
- Completude: no visualizador não está presente, pois esta função é delegada às outras aplicações do ecossistema;
- Concisão: as camadas podem ser modificadas através de filtros lógicos, visualizadas na forma de tabelas, e exportadas em formatos compatíveis com análise de dados;
- Consistência e coerência: todos os metadados da INDE são produzidos segundo o perfil de Metadados Geoespaciais do Brasil(MGB);
- Informatividade: o visualizador possui um formulário de contato que permite comunicação direta com o Diretório Brasileiro de Dados Geoespaciais(DBDG);

Na tabela 1 expandimos o que estava presente no artigo base e podemos verificar como fica a adequação do visualizador no modelo proposto, em comparação com outros três portais: dados.gov.br(Brasil), data.europa.eu(União Européia) e data.gov(EUA).

Tabela 1: Checklist para avaliar a interpretabilidade dos portais selecionados de dados governamentais abertos. Utilizamos x para indicar "Não existe" e ✓ para indicar "Existe".

	Visualizador.gov.br	Dados.gov.br	Data.europa.eu	Data.gov
Dicionários de dados	✓	✓	✓	✓
Rastreamento automático de dados	x	x	x	x
Rastreamento manual de dados	✓	✓	✓	✓
Linguagem acessível ao cidadão(tradução)	x	x	x	x
Descrições claras sobre cada conjunto de dados	✓	✓	✓	✓
Organização na interface	✓	✓	✓	✓
Metadados estruturados para cada conjunto de dados	✓	✓	✓	✓
Identificadores semânticos padronizados	x	x	✓	x
Dados em tempo real	x	x	x	x
Comunidade pública de veracidade de dados	x	x	x	✓
Equipe interna para avaliar qualidade e precisão dos dados	x	x	✓	x
Dados vinculados de outras fontes	✓	x	✓	✓
Descrições de entidades de dados vinculados	✓	x	✓	✓
Informações sobre o que está sendo realmente divulgado	✓	✓	✓	✓
Análise semântica para recuperação de dados	x	x	x	x
Linguagem padronizada e unificada	✓	x	✓	✓
Estratégia de similaridade entre vetores contextuais	x	x	x	x
Modelos ontológicos	✓	✓	✓	✓
Vocabulário controlado	✓	x	✓	✓
Metadados padronizados para cada conjunto de dados	✓	x	✓	✓
Mecanismos para sugerir dados faltantes	✓	✓	✓	✓
Medida de valor ou utilidade dos dados divulgados	✓	✓	x	x
Pesquisas de satisfação	x	x	x	x

3. Estudo de Caso: Transformando Dados Abertos em Conhecimento

3.1. O Dataset

O dataset PolRoute-DS é um conjunto de dados projetado para fomentar o desenvolvimento e a avaliação de abordagens de roteamento policial em grandes centros urbanos, que combina a estrutura espacial da cidade de interesse (no contexto deste artigo, a cidade de São Paulo), representada como um grafo conectado e direcionado de segmentos de ruas, com dados criminais obtidos de fontes públicas [6]. O dataset será combinado com uma camada de limites dos distritos da cidade de São Paulo disponibilizada no visualizador da INDE, e o objetivo desse estudo é transformar esse grande volume de dados em conhecimento compreensível em um mapa interativo.

O *dataset* é composto de 6 arquivos csv com informações sobre o tipo de ocorrência criminal, em qual bairro/distrito, segmento de rua com seus vértices, e em qual período de tempo separado por ano, mês, dia da semana, dia e período do dia. Os arquivos são: *time.csv, crime.csv, segment.csv, vertice.csv, district.csv e neighborhood.csv*. Abaixo especificamos um diagrama dono-membro [7] com as relações para guiar o resto do trabalho:

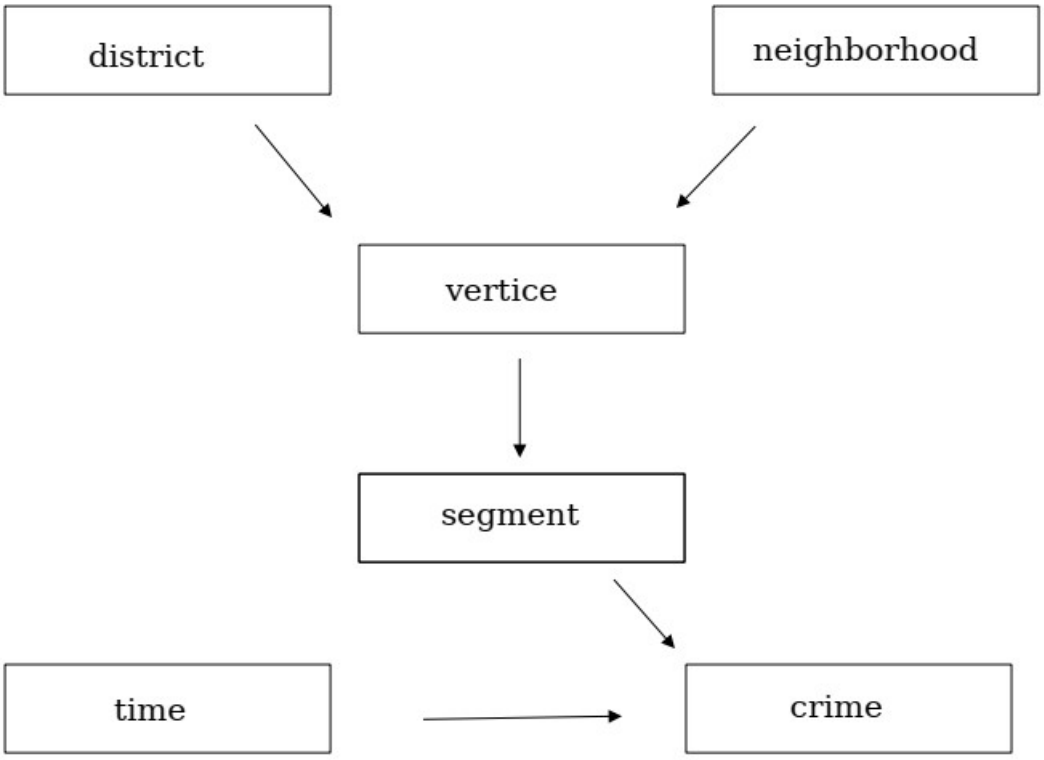

Figura 1: Diagrama dono-membro.

Vamos carregar o *dataset* no software de *datawarehouse* Apache Hive, com o objetivo de projetar consultas na linguagem HQL, uma linguagem do Hive semelhante à SQL, que possam ser processadas em paralelo e retornar em um tempo razoável um conjunto de dados que possa ser exportado para um arquivo csv. Embora o *dataset* original possua informações sobre criminalidade até para vértices e segmentos de ruas, para o escopo desse estudo, vamos considerar apenas os distritos. Analogamente, embora o *dataset* registre crimes com granularidade até o momento do dia (manhã, tarde, noite, madrugada), para o escopo deste estudo vamos considerar apenas a criminalidade por ano. Portanto, serão geradas consultas para extrair um *dataset* com os crimes por distrito e por ano. A tabela dos distritos, além de um identificador e o nome do distrito, carrega também a geometria, que será essencial para o propósito deste estudo. Como o Apache Hive opera em cima do software Apache Hadoop e seu sistema de arquivos próprio, montaremos um *cluster* de *containers* com essas aplicações, que será descrito na seção a seguir.

3.2. O Cluster

O Apache Hadoop [8] é um conjunto de utilitários de software de código aberto que facilita o uso de uma rede de muitos computadores para resolver problemas envolvendo grandes quantidades de dados e computação. Ele fornece um framework de software para o armazenamento distribuído e processamento de dados volumosos usando o modelo de programação MapReduce. O núcleo do Hadoop consiste em uma parte de armazenamento, conhecida como *Hadoop Distributed File System* (HDFS), e uma parte de processamento que é o modelo de programação *MapReduce*. Hadoop divide arquivos

em blocos grandes e os distribui entre nós em um *cluster*. Em seguida, ele transfere código empacotado para os nós para processar os dados em paralelo. Este método aproveita a localidade dos dados, onde os nós manipulam os dados aos quais têm acesso. Isso permite que o conjunto de dados seja processado de maneira mais rápida e eficiente do que em uma arquitetura de supercomputador convencional que depende de um sistema de arquivo paralelo, onde a computação e os dados são distribuídos via rede de alta velocidade [8]. Para o propósito deste trabalho vamos detalhar um pouco melhor duas partes do Hadoop: o *NameNode* e o *DataNode*

O *NameNode* está no centro do *cluster* Hadoop. Ele mantém a árvore de diretórios de todos os arquivos no sistema de arquivos e rastreia onde os dados do arquivo são realmente mantidos no *cluster*. O *DataNode*, armazena os dados reais do arquivo. Os arquivos são replicados em vários *DataNodes* em um cluster para maior confiabilidade. Especificamente, se pensarmos em termos de Hive, os dados armazenados em suas tabelas são espalhados pelos *DataNodes* dentro do *cluster* e é responsabilidade do *NameNode* rastrear esses blocos de dados [3]. Neste trabalho, para simplificar, vamos utilizar somente um *DataNode*.

O Apache Hive [9] é um projeto de software de *datawarehouse* construído sobre o Apache Hadoop para fornecer consulta e análise de dados em escala massiva. Hive oferece uma interface semelhante ao SQL para consultar dados armazenados em vários bancos de dados e sistemas de arquivos que se integram ao Hadoop. Consultas SQL tradicionais precisam ser implementadas na API Java do MapReduce para executar aplicativos e consultas SQL em dados distribuídos no entanto Hive fornece a abstração SQL necessária para integrar consultas semelhantes ao SQL (HiveQL) no Java subjacente sem a necessidade de implementar consultas na API Java de baixo nível [4]. Hive facilita a integração de linguagens de consulta baseadas em SQL com o Hadoop, que é comumente usado em aplicações de *data warehousing*.

Da arquitetura do Hive vamos especificar melhor o conceito de *MetaStore*. O Hive *MetaStore* armazena metadados para cada uma das tabelas, como esquema e localização no sistema de arquivos do Hadoop. Também inclui os metadados da partição que ajudam o driver a rastrear o progresso de vários conjuntos de dados distribuídos pelo cluster.

Vamos utilizar *containers* docker para implementar nosso cluster. Ele será constituído de 5 containers:

- namenode: operando uma imagem docker do hadoop;
- datanode: operando uma imagem docker do hadoop. Depende do namenode;
- hive-server: operando uma imagem do Hive. Depende do metastore;
- metastore: operando uma imagem do Hive. Depende do metastore-postgresql;
- metastore-postgresql: operando uma imagem do postgresql. Depende do datanode;

O *container* metastore-postgresql é utilizado para persistir os dados gerados pelo container metastore.

3.3. Carregando Os arquivos no Apache Hive

Dentro de uma pasta foram criados os arquivos hadoop-hive.env, com as configurações para o namenode, datanode e hive-server, e docker-compose.yml, com as configurações necessárias para subir o 5 *containers* do nosso *cluster*. Nesse mesmo diretório também foi criada a pasta polroute, aonde deve ser descompactado o dataset, e foram criados dois arquivos HQL. Um contém o DDL das tabelas sem fragmentação(database_ddl_nofrag.hql), o outro contém o DDL das tabelas fragmentadas e os comandos de carga de dados para as tabelas com FHD(database_ddl_frag_partition.hql). Com essa estrutura de diretórios e arquivos configurada, podemos rodar no terminal o comando abaixo para subir o *cluster*:

sudo docker compose up

Com o *cluster* rodando, podemos abrir um outro terminal e entrar na linha de comando do *container* hive-server:

sudo docker exec -it hive-server /bin/bash

Agora podemos navegar ao diretório polroute, e executar os arquivo HQL com o DDL das tabelas. No exemplo abaixo vamos carregar as tabelas não fragmentadas:

cd ..

cd polroute

hive -f database_ddl_nofrag.hql

Com as tabelas criadas podemos dar carga nelas a partir dos dados de cada um dos arquivos csv presentes no diretório polroute:

hadoop fs -put crime.csv hdfs://namenode:8020/user/hive/warehouse/trabalho_egov.db/crime

hadoop fs -put time.csv hdfs://namenode:8020/user/hive/warehouse/trabalho_egov.db/time

hadoop fs -put district.csv hdfs://namenode:8020/user/hive/warehouse/trabalho_egov.db/district

hadoop fs -put neighborhood.csv hdfs://namenode:8020/user/hive/warehouse/trabalho_egov.db/neighborhood

hadoop fs -put vertice.csv hdfs://namenode:8020/user/hive/warehouse/trabalho_egov.db/vertice

hadoop fs -put segment.csv hdfs://namenode:8020/user/hive/warehouse/trabalho_egov.db/segment

Não devemos nos surpreender com o fato dessa carga ser extremamente rápida(mais ou menos 1 segundo) mesmo para os maiores arquivos. O que o hadoop faz é apenas copiar o arquivo informado para a sua localização dentro do HDFS. O Apache Hive também não faz nenhuma validação dos dados no momento da escrita, somente na leitura (*schema on-read*).

Com as tabelas criadas e preenchidas, podemos acessar a linha de comando do Hive e começar a rodar nossas queries.

hive

use trabalho_egov

Vamos ter consultas para gerar um dataset com todos os crimes, por distrito, dos anos de 2011 a 2018, sem a geometria para deixar o arquivo menor. Para termos a geomtria disponível, vamos gerar datasets análogos porém ano a ano, um dataset separado para cada ano de 2011 a 2018, com os crimes do distrito e a geometria do distrito. Abaixo temos a query para gerar o dataset principal, sem a geometria, e em seguida um exemplo da query para gerar o dataset com os crimes, distrito e geometria para o ano de 2011, as consultas para os outros anos são análogas.

```
INSERT INTO districts_crimes_allyears_nogeom
select
    district_frag_geom.id as id,
    district_frag_geom.name as name,
    time_frag.year as year,
    sum(crime.total_feminicide) as feminicide,
    sum(crime.total_homicide) as homicide,
    sum(crime.total_felony_murder) as felony_murder,
    sum(crime.total_bodily_harm) as bodily_harm,
    sum(crime.total_theft_cellphone) as theft_cellphone,
    sum(crime.total_armed_robbery_cellphone) as robbery_cellphone,
    sum(crime.total_theft_auto) as theft_auto,
    sum(crime.total_armed_robbery_auto) as armed_robbery_auto,
    sum(crime.total_feminicide) +
        sum(crime.total_homicide) +
        sum(crime.total_felony_murder) +
        sum(crime.total_bodily_harm) +
        sum(crime.total_theft_cellphone) +
        sum(crime.total_armed_robbery_cellphone) +
        sum(crime.total_theft_auto) +
        sum(crime.total_armed_robbery_auto) as criminal_index
from
    vertice, district_frag_geom, segment, crime, time_frag

where time_frag.id = crime.time_id
    and crime.segment_id = segment.id
                and    (segment.start_vertice_id    =    vertice.id    OR
segment.start_vertice_id = vertice.id)
    and vertice.district_id = district_frag_geom.id
     and time_frag.year IN (2011, 2012, 2013, 2014, 2015, 2016, 2017,
2018)
```

```
                group by district_frag_geom.id, district_frag_geom.name,
time_frag.year
          order by district_frag_geom.name ASC, time_frag.year DESC
      ;

      INSERT INTO districts_crimes_2011_geom
      select
          distrito.*
          , geometria.geometry
      from
          (select
          district_frag_geom.id as id,
          district_frag_geom.name as name,
          time_frag.year as year,
          sum(crime.total_feminicide) as feminicide,
          sum(crime.total_homicide) as homicide,
          sum(crime.total_felony_murder) as felony_murder,
          sum(crime.total_bodily_harm) as bodily_harm,
          sum(crime.total_theft_cellphone) as theft_cellphone,
          sum(crime.total_armed_robbery_cellphone) as robbery_cellphone,
          sum(crime.total_theft_auto) as theft_auto,
          sum(crime.total_armed_robbery_auto) as armed_robbery_auto,
          sum(crime.total_feminicide) +
              sum(crime.total_homicide) +
              sum(crime.total_felony_murder) +
              sum(crime.total_bodily_harm) +
              sum(crime.total_theft_cellphone) +
              sum(crime.total_armed_robbery_cellphone) +
              sum(crime.total_theft_auto) +
              sum(crime.total_armed_robbery_auto) as criminal_index
      from
          vertice, district_frag_geom, segment, crime, time_frag

      where time_frag.id = crime.time_id
          and crime.segment_id = segment.id
                      and    (segment.start_vertice_id    =    vertice.id    OR
segment.start_vertice_id = vertice.id)
          and vertice.district_id = district_frag_geom.id
          and time_frag.year = 2011
                group by district_frag_geom.id, district_frag_geom.name,
time_frag.year)
          as distrito,
          (select
```

```
            district_frag_geom.id,
            district_frag_geom.geometry
            from district_frag_geom)
        as geometria
    where distrito.id = geometria.id
    ;
```

Essas consultas foram usadas para criar tabelas separadas com o resultado de cada uma, o motivo é o apache hive criar um arquivo para cada tabela, precisamos de todos os dados uma só tabela para poder gerar um arquivo csv não fragmentado, único.

3.4. Exportando os datasets

Com as tabelas criadas, podemos gerar um arquivo csv com seu conteúdo com o comando:

hadoop fs -cat hdfs://namenode:8020/user/hive/warehouse/trabalho_egov.db/districts_crimes_all_years_no_geom/* > crimes_all_years.csv

É preciso ficar atento para não gerar o arquivo csv com o mesmo nome da tabela, pois será gerado um erro e o arquivo não será criado. Para as tabelas com os crimes por distrito com a geometria, o comando é análogo:

hadoop fs -cat hdfs://namenode:8020/user/hive/warehouse/trabalho_egov .db/districts_crimes_2011_geom/* > crimes_2011.csv

3.5. Convertendo a geometria

A geometria do dataset original, e presente agora nos datasets dos arquivos csv que exportamos do Apache Hive, está no formato EWKB(*Extended Well Know Binary*), um formato não suportado pela biblioteca de mapa que vamos usar para mostrar as camadas, o MapLibre. Portanto usaremos o software QGIS(Quantum GIS) para converter para o formato Lat/Long na projeção EPSG:4326.

Após abrir o QGIS, devemos navegar até o menu Layer > Add Layer > Add/Edit Virtual Layer. Na seção embedded layer, devemos localizar o arquivo csv com a geometria, e na seção query vamos colar uma consulta que converte o campo da geometria, e adiciona ela e os outros campos do csv na camada.

```
select
    GeomFromEWKB(geometry) as geom,
      id, name, year, feminicide, homicide, felony_murder, bodily_harm,
theft_cellphone, robbery_cellphone, theft_auto, robbery_auto, criminal_index
from crimes_2011_csv
```

Após isso, clicar em Add e fechar a janela. A camada foi adicionada ao projeto. Caso ela não apareça, clique com o botão direito na camada e escolha a opção “Zoom to Layer(s)”. Para exporta a camada para o formato GeoJSON, clique com o botão direito, e escolha “Export”, escolha o formato GeoJSON, nomeie o arquivo e pronto.

3.6. Visualizando o dataset no mapa

Utilizaremos a biblioteca JavaScript MapLibre para carregar nosso mapa. Junto do dataset também vamos carregar no mapa uma camada disponibilizada no visualizador da inde com os limites dos distritos de São Paulo, o motivo vai ficar claro mais a frente. Após criar o mapa, para adicionar a camada o dataset de crimes com geomtria fazemos:

```
map.on('load', () => {
  map.addSource('source_crimes', {
    'type': 'geojson',
    'data': crimes_2011.geojson
  });
  map.addLayer({
    'id': 'layer_crimes',
    'type': 'fill',
    'source': 'source_crimes',
    'layout': {},
    'paint': {
      'fill-color': [
        'interpolate',
        ['linear'],
        ['get', ‘criminal_index’],
        0, colors[0],
        colorStops[1], colors[1],
        colorStops[2], colors[2],
        colorStops[3], colors[3],
        colorStops[4], colors[4],
        colorStops[5], colors[5],
        colorStops[6], colors[6],
        colorStops[7], colors[7],
        colorStops[8], colors[8]
      ],
      'fill-opacity': 1
    }
  });
```

E para adicionar a camada com a geometria dos distritos do estado de São Paulo obtida pelo visualizador da INDE:

```
  map.addSource('limites_dos_distritos', {
```

```
        'type': 'geojson',
        'data': ‘https://ide.emplasa.sp.gov.br/geoserver/ows?
service=WMS&version=1.1.0&request=GetFeature&typeName=emplasa:LIMITES_E
MPLASA_DISTRITO_UIT_MSP&styles=&tiled=true&srsName=EPSG:4326&format
=application/geojson&transparent=true’
    });

    map.addLayer({
        'id': 'limites_dos_distritos',
        'type': 'line',
        'source': 'limites_dos_distritos',
        'layout': {},
        'paint': {
            'line-color': '#000000'
        }
    });
```

O resultado pode ser visto na figura 3.

Com os *datasets* gerados podemos também obter outros tipos de visualizações para analisarmos os dados, como por exemplo a evolução do índice criminal por ano(Figura 4), o total de cada crime considerado no dataset por ano(Figura 5) e até uma matriz de calor para o ano mais violento, 2017(Figura 6). Por último, aumentando bastante o zoom do mapa em algumas áreas, podemos constatar que a geometria dos dados do *dataset* original, está escapando dos limites da geometria dos distritos obtida do Visualizador da INDE(Figura 7). Fomos capazes de descobrir essa limitação/erro graças à p interoperabilidade dos dados de diferentes fontes.

Neste artigo apresentamos uma análise da interpretabilidade do portal de dados abertos geoespaciais do Visualizador da Infraestrutura Nacional de Dados Espaciais, aonde pudemos atestar seus pontos fortes e suas fraquezas. A importância dos dados abertos foi verificada com um estudo de caso, aonde foi essencial a interoperabilidade entre os provedores de dados. Um dado aberto obtido de um artigo acadêmico, foi combinado com dados obtidos da INDE, através de um longo processo de gerenciamento de grandes volumes de dados, de forma a obter informações úteis.

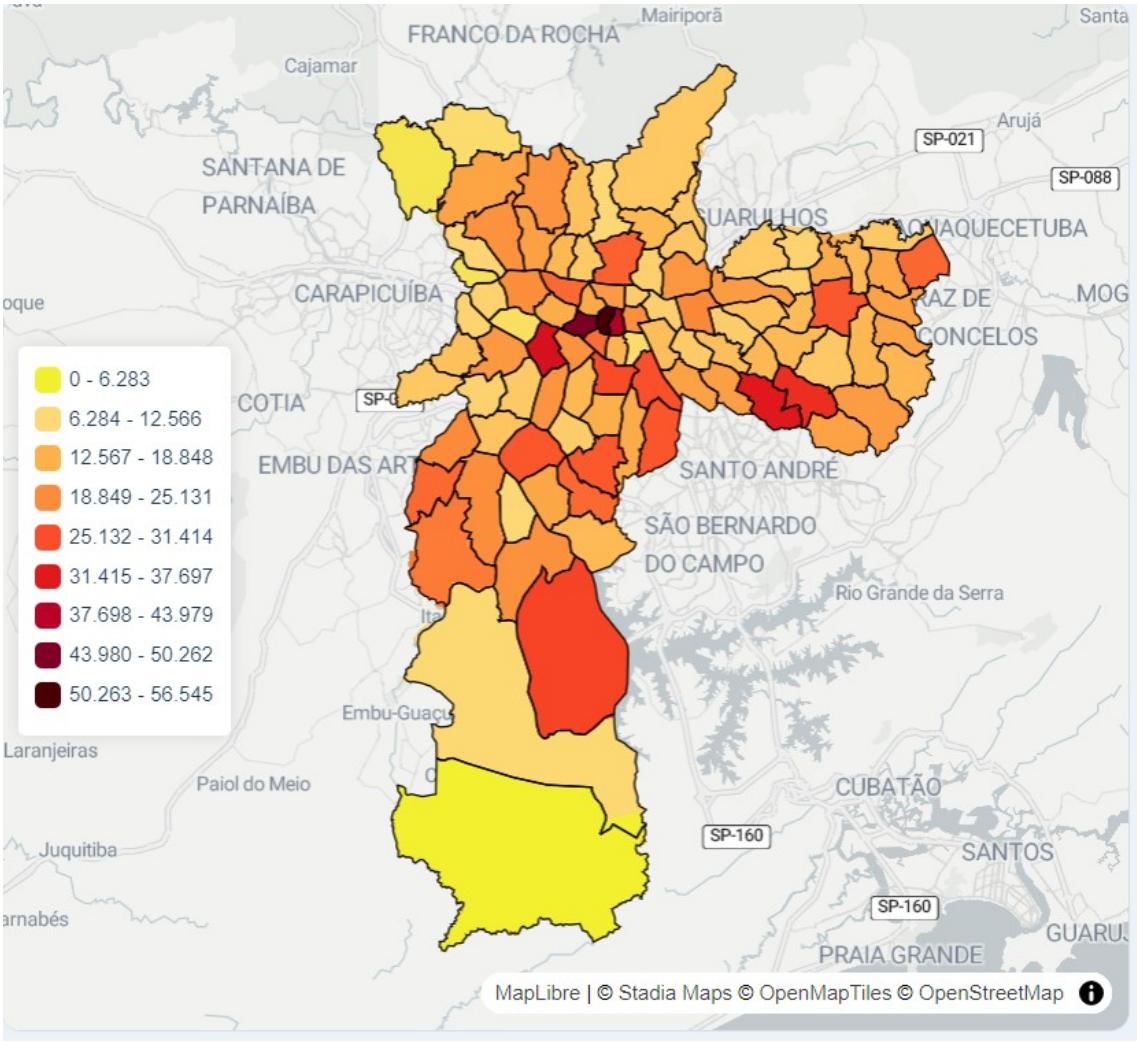

Figura 3: índice criminal dos distritos do município de São Paulo em 2017.

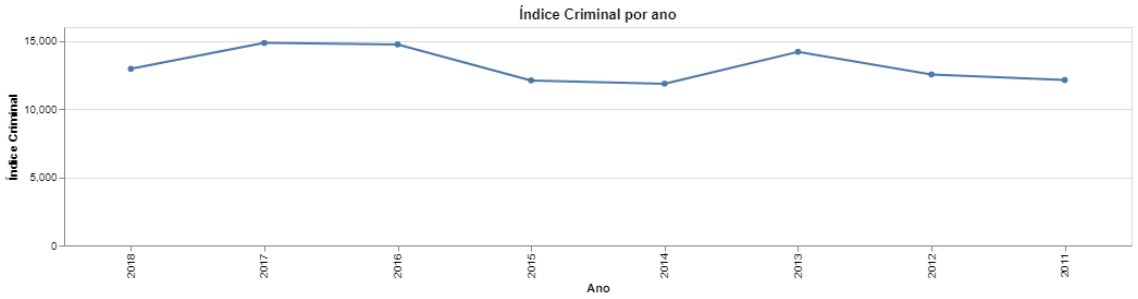

Figura 4: evolução do índice criminal.

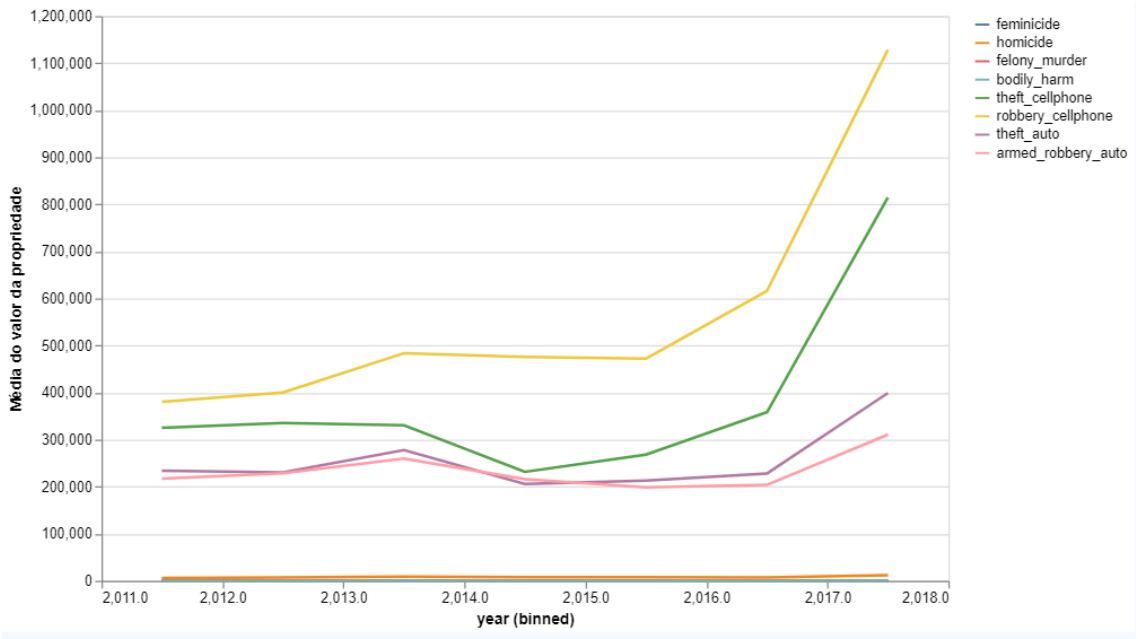

Figura 5: total de cada crime por ano.

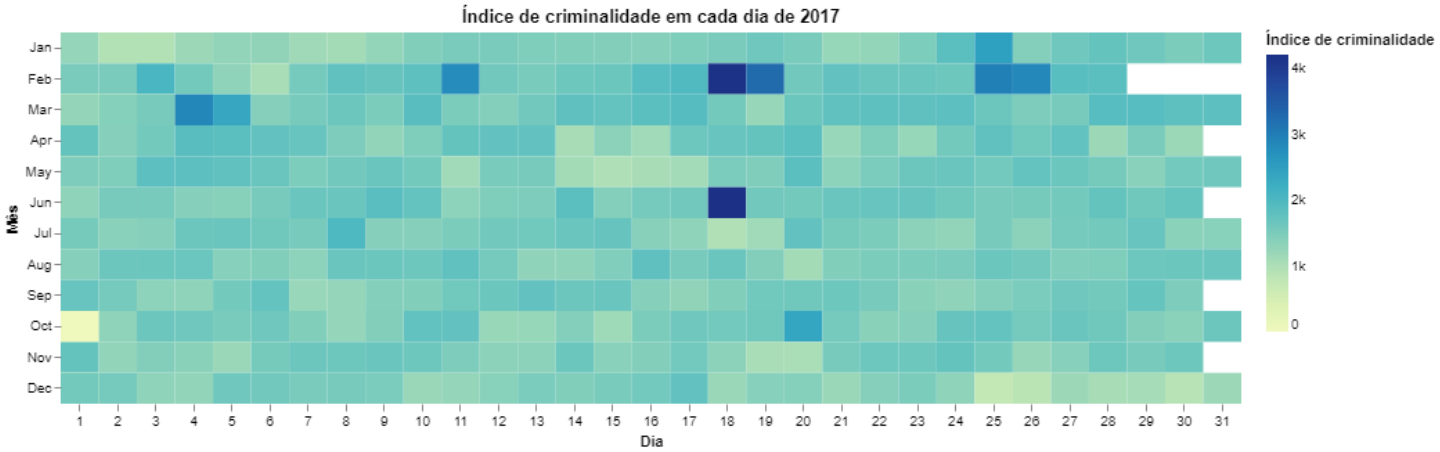

Figura 6: matriz de calor com os crimes por dia em 2017.

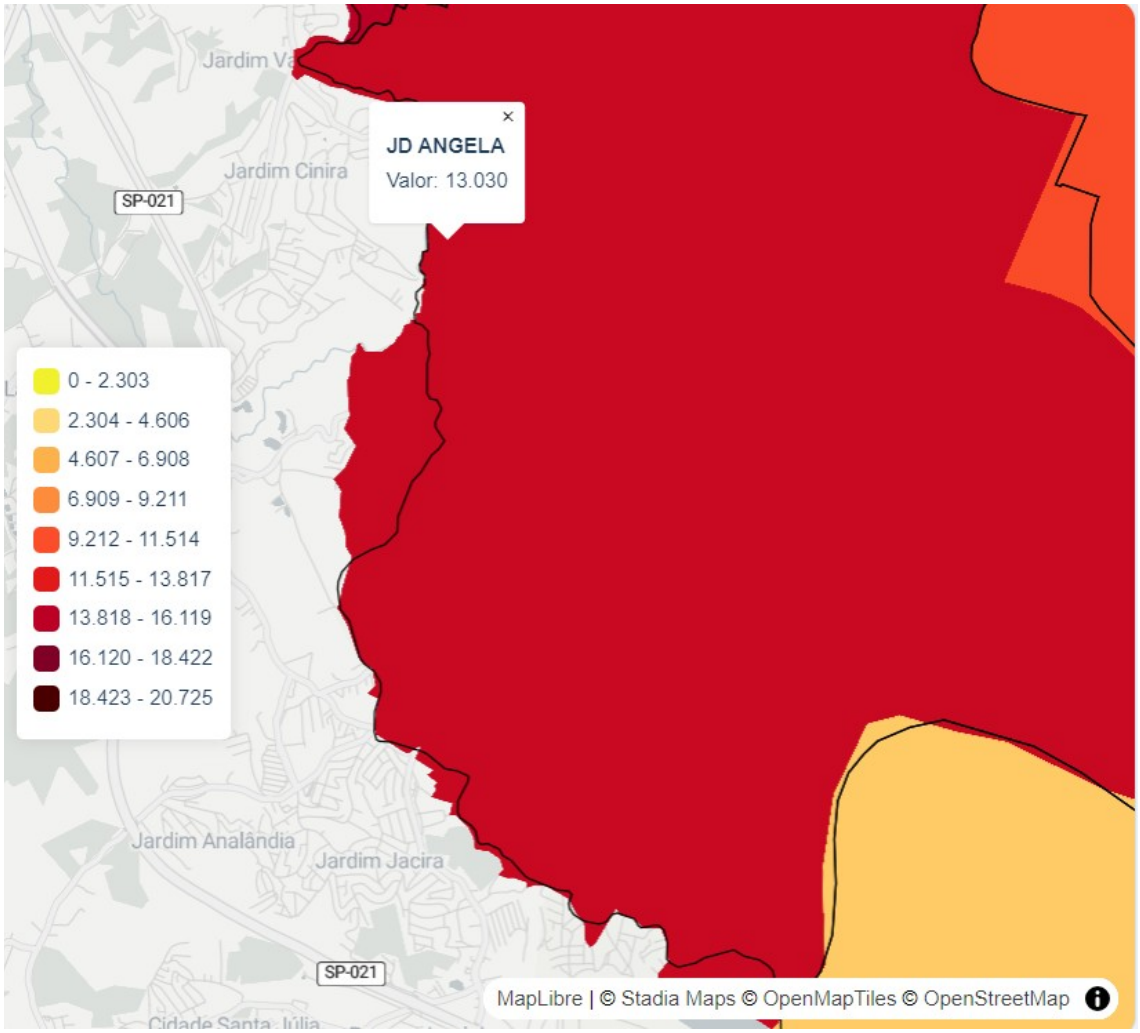

Figura 7: erros na geometria do dataset PolRoute.

Referências

[1] Raissa Barcellos, Flavia Bernardini, José Viterbo, Towards defining data interpretability in open data portals: Challenges and research opportunities, Information Systems, Volume 106, 2022, 101961, ISSN 0306-4379, https://doi.org/10.1016/j.is.2021.101961.

[2](2008, November 27). *Decreto 666/2008*. Planalto.gov.br. Retrieved July 4, 2024, from https://www.planalto.gov.br/ccivil_03/_ato2007-2010/2008/decreto/d6666.htm

[3] (2008, November 27). *Catálogo de Metadados INDE*. INDE. Retrieved July 4, 2024, from https://inde.gov.br/CatalogoMetadados

[4] (2008, November 27). *Catálogo de Geosserviços*. INDE. Retrieved July 4, 2024, from https://inde.gov.br/CatalogoGeoservicos

[5] (2008, November 27). *Visualizador de Mapas*. INDE. Retrieved July 4, 2024, from https://inde.gov.br/Visualizador

[6] Cunha Sá, B., Muller, G., Banni, M., Santos, W., Lage, M., Rosseti, I., Frota, Y. and de Oliveira, D. 2022. PolRoute-DS: a Crime Dataset for Optimization-based Police Patrol Routing. Journal of Information and Data Management. 13, 1 (Aug. 2022). DOI:https://doi.org/10.5753/jidm.2022.2355.

[7] Özsu, M. Tamer, Valduriez, P. (2011) “Principles of Distributed Database Systems - Third Edition”, p.81-98. Springer

[8] Apache Hadoop. (2024, June 28). In *Wikipedia*. https://en.wikipedia.org/wiki/Apache_Hadoop

[9] Apache Hive. (2024, June 28). In *Wikipedia*. https://en.wikipedia.org/wiki/Apache_Hive