UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ANDREZA FERNANDES DE OLIVEIRA

UMA ARQUITETURA E IMPLEMENTAÇÃO DO MÓDULO DE VISUALIZAÇÃO PARA BIBLIOTECA PYMOVE

FORTALEZA

2019

ANDREZA FERNANDES DE OLIVEIRA

UMA ARQUITETURA E IMPLEMENTAÇÃO DO MÓDULO DE VISUALIZAÇÃO PARA BIBLIOTECA PYMOVE

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientador: Prof. Dr. José Antônio Fernandes Macedo

Coorientador: Prof. Dr. Regis Pires Magalhães

FORTALEZA

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

O45a Oliveira, Andreza Fernandes de.
Uma arquitetura e implementação do módulo de visualização para biblioteca PyMove / Andreza Fernandes de Oliveira. – 2019.
64 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Ciências, Curso de Computação, Fortaleza, 2019.
Orientação: Prof. Dr. José Antônio Fernandes Macedo.
Coorientação: Prof. Dr. Regis Pires Magalhães.

1. Trajetórias. 2. Análise de trajetórias. 3. Mobilidade. 4. Visualização de dados. I. Título.
CDD 005

ANDREZA FERNANDES DE OLIVEIRA

UMA ARQUITETURA E IMPLEMENTAÇÃO DO MÓDULO DE VISUALIZAÇÃO PARA BIBLIOTECA PYMOVE

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. José Antônio Fernandes Macedo (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Regis Pires Magalhães (Coorientador)
Universidade Federal do Ceará (UFC)

Prof[a]. Ma. Lívia Almada Cruz Rafael
Universidade Federal do Ceará (UFC)

Me. Nicksson Ckayo Arrais de Freitas
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

Inicialmente agradeço à Deus pelas oportunidades e aprendizados concedidos ao longo da minha curta existência que me permitem tornar-se uma pessoa melhor. Em seguida, agradeço aos meus pais e minha família, por serem meu apoio. Em especial a minha mãe e minha vozinha, que nunca mediram esforços para propiciar uma vida mais digna a meus irmãos e a mim, nos incentivando sempre a conquistar tudo com nossos esforços e a seguir sempre pelo caminho da educação.

Sou bastante grata ao Prof. Dr. José Antônio Fernandes Macedo e toda a equipe do Insight Data Science Lab pela oportunidade de integrar essa família e estar ao lado de pessoas perspicazes e cheias de garra perante aos desafios.

Agradeço também ao Regis Pires Magalhães, Nicksson Arrais, Lívia Almada, Francisco Carlos Júnior e Lucas Peres pelas participações nas reuniões para o desenvolvimento deste trabalho, pelas sugestões e pela disponibilidade.

Agradeço aos professores Regis Pires, Lívia Almada e Ticiana Linhares pela participação da banca avaliadora deste trabalho.

Aproveito aqui para agradecer todos aqueles que fizeram parte da minha trajetória nessa universidade: os funcionários do departamento, os professores, os meus companheiros do PET Computação, e todos aqueles que me agregaram e que eu agreguei, que proporcionaram troca de conhecimentos não só profissional, mas também de alma. Um abraço especial para minha querida amiga Arina Sanches, por sempre estar disposta a enfrentar as lutas que a vida propicia ao meu lado, inclusive no desenvolvimento deste trabalho!

E por último, não menos importante, ao meu companheiro Edson Pinheiro, por sempre estar ao meu lado, me motivar a dar sempre o meu melhor, a me amparar nas horas que mais preciso e não me deixar desistir.

Serei eternamente grata à todos vocês que contribuíram e contribuem para minha evolução. Obrigada por me fazerem sentir mais forte e mais feliz todos os dias!

"Vocês que fazem parte dessa massa
Que passa nos projetos do futuro
É duro tanto ter que caminhar
E dar muito mais do que receber
E ter que demonstrar sua coragem
À margem do que possa parecer
E ver que toda essa engrenagem
Já sente a ferrugem lhe comer"

(Zé Ramalho)

RESUMO

O volume de dados de trajetória vem crescendo de forma acelerada em razão de diversos fatores, incluindo o avanço e redução dos custos de tecnologias de localização, como sensores *Global Positioning System* (GPS). Esta realidade destaca a necessidade do desenvolvimento de novas tecnologias e técnicas computacionais para descobrir conhecimentos e informações a partir desses dados. Embora o grande número de publicações de técnicas sobre o tema, é notável a ausência de softwares e ferramentas que ofereçam suporte e possam auxiliar pesquisadores a analisar e manipular esse tipo de dado de forma centralizada. Este trabalho traz como proposta uma modelagem e arquitetura de funções previamente existentes para a biblioteca PyMove, voltada para o trabalho com dados espaço-temporais, através da arquitetura e modelagem. Além disso, este trabalho também apresenta a implementação de técnicas de visualização para agregar o módulo de visualizações desta biblioteca. No final deste trabalho é possível fazer um apanhado geral e identificar pontos relevantes em relação aos trabalhos relacionados analisados, como as vantagens, tais como extensibilidade, possibilidade de escalabilidade, e desvantagens, como possuir um escopo reduzido de operações, assim como as visualizações produzidas.

Palavras-chave: Trajetórias. Análise de trajetórias. Mobilidade. Visualização de dados.

ABSTRACT

Trajectory data volume is growing rapidly due to a number of factors, including the advancement and cost savings of location technologies such as GPS sensors. This reality highlights the need for the development of new technologies and computational techniques to discover knowledge and information from this data. Despite the large number of technical publications on the subject, it is notable the absence of software and tools that support and can help researchers to analyze and manipulate this type of data in a centralized manner. This work proposes a modeling and architecture of previously existing functions for the PyMove library, focused on working with spatiotemporal data, through architecture and modeling. In addition, this paper also presents the implementation of visualization techniques to aggregate the visualization module of this library. At the end of this paper it is possible to make an overview and identify relevant points in relation to the related works analyzed, such as advantages such as extensibility, scalability, and disadvantages, such as having a reduced scope of operations, as well as the visualizations produced.

Keywords: Trajectories. Trajectory Analysis. Mobility. Data visualization.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
GPS	*Global Positioning System*
HTML	*HyperText Markup Language*
KDE	Kernel Density Estimation
KNN	K-Nearest Neighbors
PEP	*Python Enhancement Proposal*
RAM	*Random Access Memory*

SUMÁRIO

1 INTRODUÇÃO

O volume de dados relacionados a trajetórias teve um grande aumento em razão do avanço de tecnologias de localização e a redução dos seus custos, como sensores GPS. O acúmulo desses dados, junto à possibilidade de extração de mais informações, proporciona o desenvolvimento de áreas de estudos, como Geografia, Sociologia e Ciência da Computação (ZHENG, 2015).

Nesse contexto, há a forte motivação para a busca de técnicas e ferramentas apropriadas para extrair conhecimento dessas coleções de dados. A análise desses dados utilizando técnicas exploradas pela área de mineração de dados adquirem uma maior relevância, por englobar os processos de seleção, limpeza e transformação que permitem extrair informações sobre esses dados. Porém, automatizar completamente o processo de análise dados pode não abranger peculiaridades dos dados.

Frente a isso, a etapa de visualização exploratória dos dados admite que os indivíduos possam incorporar o seu conhecimento do domínio. Logo, é possível criar associações, facilitar o entendimento dos dados, auxiliar na tomada de decisões, detectar padrões e descrever detalhes (ANDRIENKO; ANDRIENKO, 2013).

Como apresentado em Zheng (2015), diversas pesquisas são realizadas no campo de estudo de dados de trajetórias. No entanto, mesmo diante do grande volume de trabalhos, nota-se a ausência de softwares e ferramentas que possam auxiliar pesquisadores a lidar e extrair as informações que esses dados possam oferecer.

1.1 Objetivos

A partir dessas necessidades, o objetivo principal deste trabalho é propor uma biblioteca extensível para mineração de dados de trajetória e implementar operações para o módulo de visualização desta biblioteca. Isso é possível através da modelagem de funções previamente existentes da biblioteca PyMove[1], desenvolvida na linguagem de programação Python, direcionada para a manipulação de dados espaço-temporais, com foco em trajetórias, disponível para Python, desenvolvida por mestrandos e doutorandos do Insight Data Science Lab[2], como presente em (MAGALHÃES, 2018).

Em relação as visualizações, o presente trabalho também busca oferecer formas de

[1] Biblioteca disponível em: https://github.com/InsightLab/PyMove
[2] https://www.insightlab.ufc.br/

visualização de dados de trajetória e mobilidade, através do uso de bibliotecas para geração de gráficos disponíveis em Python.

Os objetivos específicos deste trabalho consistem em:

- Realizar uma revisão bibliográfica sobre a análise e visualização de dados de trajetórias, investigando os conceitos fundamentais e técnicas de visualização para esse tipo de dado. Além disso, estudar as características de trabalhos com objetivos parecidos com o proposto neste trabalho;
- Elaborar e estruturar em módulos as operações, previamente existentes, da biblioteca;
- Verificar e validar as operações da biblioteca, através do uso de ferramentas e seguindo guias de estilo;
- Prover à biblioteca características como extensibilidade e familiaridade de sintaxe a biblioteca Pandas, Seção 2.3.1, amplamente utilizada para manipulação de dados;
- Elaborar visualizações a partir de dados brutos de trajetórias;
- Apresentar os benefícios e as limitações da biblioteca desenvolvida.

1.2 Contribuições

A principal contribuição deste trabalho é a arquitetura e modelagem proposta para a biblioteca, o estudo sobre técnicas de visualização de trajetórias e a implementação e adição dessas técnicas ao módulo de visualizações do PyMove.

1.3 Estrutura do trabalho

Este trabalho está organizado da seguinte forma: no Capítulo 2 estão descritos todos os conceitos necessários para o entendimento desse trabalho; no Capítulo 3 são descritos os trabalhos encontrados relacionados ao presente trabalho, bem como um comparativo entre eles; no Capítulo 4 são apresentados os detalhes de cada etapa executada nesse trabalho; por fim, no Capítulo 5 são apresentadas as conclusões obtidas com o desenvolvimento deste trabalho, tal como as expectativas para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados conceitos e temas importantes para o entendimento deste documento. Estes estendem-se às técnicas e abordagens utilizadas no trabalho com dados espaço-temporais, abrangendo a fase de representação dos dados, técnicas de pré-processamento, modelos e, por fim, as técnicas adotadas para a visualização desses dados. Além disso, também são apresentados as diretrizes e conjunto de ferramentas utilizados para a elaboração do trabalho.

2.1 Trajetórias

Em Zheng (2015), uma trajetória é definida como um traço gerado por um objeto em movimento no espaço geográfico, geralmente representado por uma série de pontos cronologicamente ordenados, por exemplo, $p_1 \rightarrow p_2 \rightarrow ... \rightarrow p_n$, sendo n a quantidade de pontos de uma trajetória. Cada ponto consiste em coordenadas geoespaciais (latitude e longitude) e uma marcação de tempo. Esses pontos também podem possuir atributos como velocidade e o tipo do objeto que realiza tal trajetória.

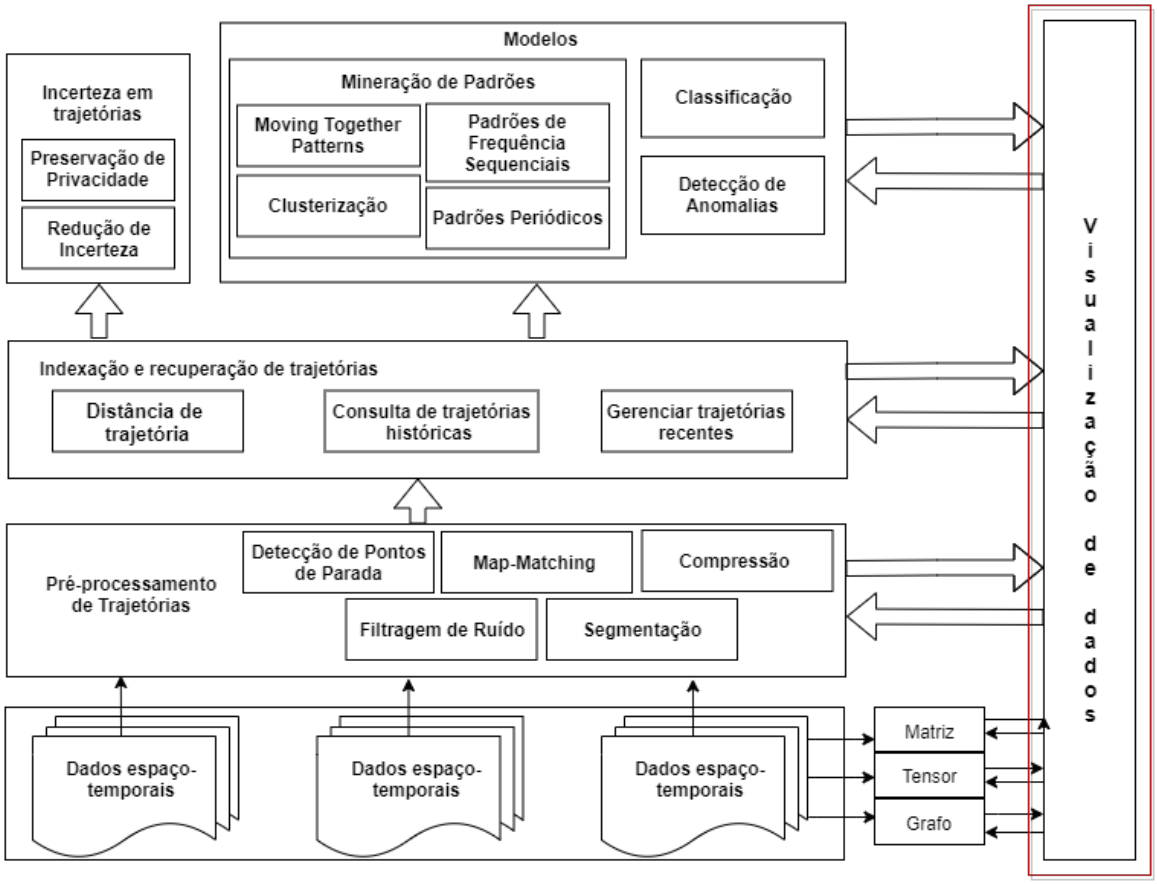
Na literatura, diversos trabalhos que utilizam técnicas de mineração de dados espaço-temporais têm sido realizados: Batista *et al.* (2011) descreve o desenvolvimento de uma aplicação para monitoramento de vegetação aquática; Morshed *et al.* (2019) propõe um sistema para análise visual de trajetórias de crime; em Sacharidis *et al.* (2008) é relatado o uso da mineração de dados para descoberta de rotas com grandes volumes de tráfego, também conhecidas como rotas quentes, as quais podem ajudar autarquias de trânsito na identificação de problemas e auxiliar na tomada de decisões. Além disso, também existem aplicações outras finalidades, como facilitar a locomoção, por exemplo o Google Maps[1] e Waze[2], monitoramento espaço-temporal, redes sociais, como abordado em Karimi *et al.* (2013), sistemas de transporte inteligentes e computação urbana (ZHENG *et al.*, 2014).

Para trabalhar com dados espaço-temporais e realizar análises, é necessária a aplicação uma série de etapas, que podem ser agrupadas em categorias e áreas de estudo. A Figura 1 é uma adaptação das etapas e processos destacados por Zheng (2015). Essa adaptação delimita-se a adequação do escopo e o foco deste trabalho, como a adição da etapa de visualização de dados. Nesta figura encontra-se em destaque a etapa de Visualização de dados por ser o foco deste trabalho.

[1] https://www.google.com.br/maps
[2] https://www.waze.com/pt-BR

Neste trabalho são explorados temas ligados as etapas de trabalho com os dados de trajetórias: as formas para representação desses dados; técnicas de pré-processamento, para efetuar a limpeza e manipulação prévia dos dados; técnicas para indexação e recuperação de trajetórias; modelos, que agrupam as áreas de incerteza (*uncertainty*), mineração de padrões, classificação e detecção de anomalias; e com aprofundamento as técnicas direcionadas para visualização dos dados de trajetórias. Nas seções abaixo serão descritas superficialmente essas etapas, com foco na etapa de visualização de dados.

Figura 1 – Paradigmas presentes no processo de mineração de dados de trajetórias.



Fonte: adaptação de Zheng (2015)

2.1.1 Representação dos dados de trajetória

As trajetórias podem ser representadas em outras estruturas de dados, além da sua forma original. As diferentes estruturas trazem aplicações que enriquecem as metodologias para extração de conhecimento. Abaixo estão citadas as principais formas de representação desses dados, conforme destacado em (ZHENG, 2015):

- **Grafos**: o principal esforço para a transformação dos dados de trajetórias para este tipo de representação é a delimitação do que são os nós e as arestas do grafo. Os métodos para realizar essa transformação variam de acordo com o problema tratado e definido. Por exemplo, a rede rodoviária é essencialmente um gráfico direcionado, onde um nó é um cruzamento e uma aresta indica um segmento de estrada. Pode-se atribuir pesos a essas arestas relacionados ao volume do tráfego, por exemplo. Portanto, é uma das abordagens mais intuitivas de transformação dos dados de trajetória para esse tipo de representação. Pode-se também construir grafos seguindo abordagens de referência e região, conforme descrito em (ZHENG, 2015).
- **Matriz**: essa representação pode ajudar a complementar informações ausentes e/ou identificar anomalias. Para realizar essa transformação é necessário identificar o significado de uma linha, da coluna e o valor da célula.
- **Tensor**: é uma extensão da representação em matriz. A transformação segue os passos da transformação em matriz, com a adição de uma terceira dimensão para acomodar informação adicional.

2.1.2 Pré-processamento

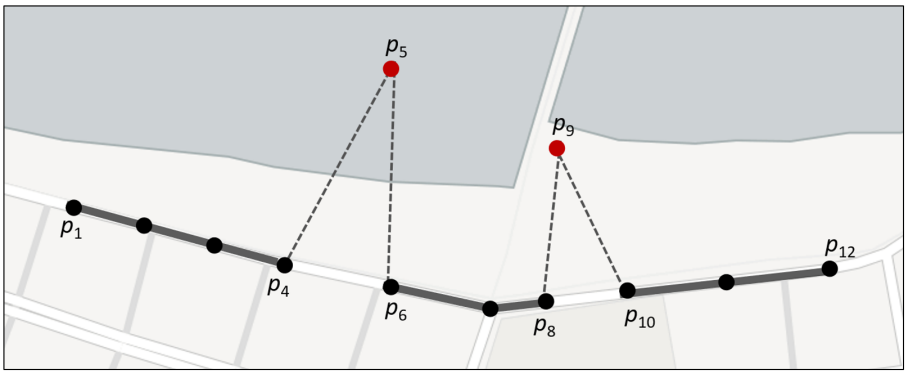
Antes de iniciar a análise dos dados, é importante e necessário realizar a etapa de pré-processamento para evitar problemas que inviabilize as análises.

O grande número de pontos pode trazer problemas, como elevação do custo computacional. Além disso, o uso de sensores pode acarretar ruídos aos dados brutos, ou até mesmo dados errôneos. Os ruídos nos dados também podem ser advindos de forma proposital, através de técnicas de privatização de dados, nas quais as pessoas não desejam revelar seus dados de localização sendo necessário fazer uma aproximação de posicionamento. Por isso, é necessário a etapa de pré-processamento dos dados, com a finalidade de solucionar e/ou minimizar tais problemas, evitando assim poluição visual durante a análise e visualização dos dados.

Esta etapa consiste nos processos de transformações, correções, detecção de anomalias e pontos de parada, compressão dos dados, remoção ou redução dos erros de posicionamento, encontrar posições que estejam faltantes, dentre outras possíveis modificações nos dados, com a finalidade de tratar essas diversas vertentes sem perder as características da trajetória. Por exemplo, posições faltantes podem ser descobertas via interpolação polinomial dos dados (HOREMUŽ; ANDERSSON, 2006).

A Figura 2 apresenta um exemplo de dados possivelmente errôneos no contexto de uma trajetória. Esse erro pode ser ocasionado por um possível ruído no sensor GPS, acarretando a falha na captura dos pontos p_5 e p_9. Esses pontos, no entanto, conforme destacado em Gomes *et al.* (2018), podem ser detectados através do cálculo da média e desvio padrão para detectar velocidades com grande discrepância em relação aos demais pontos e contexto.

Figura 2 – Trajetória com pontos, possivelmente, errôneos.

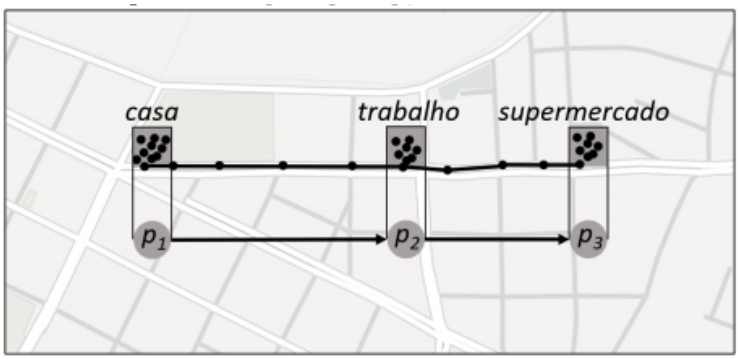


Fonte: (Gomes *et al.*, 2018)

Em Zheng (2015) são apresentadas cinco técnicas básicas para processar dados de trajetória. As técnicas são descritas em seguida.

1. **Filtragem de ruído**: realiza a remoção de dados com distorção ou erros devido aos ruídos, geralmente causados por falha na leitura do sensor de GPS.
   Analisando a Figura 2, a remoção dos pontos p_5 e p_9 pode ser realizada aplicando técnicas de filtragem de ruídos.
2. **Detecção de pontos de paradas**: possui o objetivo de identificar os locais onde o objeto ou indivíduo permaneceu parado, por algum limite de tempo, durante a sua trajetória. Esses pontos podem fornecer a compreensão do movimento de um indivíduo, possibilitando a representação de uma trajetória apenas pelos lugares mais frequentes.
   A Figura 3 mostra uma trajetória, que após executado um algoritmo para detectar pontos de parada, são encontrados três pontos de parada: a casa, trabalho e supermercado. Esses pontos podem representar essa trajetória, como pontos p_1, p_2 e p_3, respectivamente.
3. **Compressão**: possui o objetivo de reduzir o tamanho dos dados de trajetória através de um conjunto de técnicas, a fim de evitar sobrecarga de processamento e minimizar a carga no armazenamento de dados, gerando uma representação mais compactada que descreve a trajetória de um objeto. Conforme Gomes *et al.* (2018), do ponto de vista geométrico, os algoritmos de compressão buscam a simplificação das formas, com o objetivo de remover

Figura 3 – Resultado da execução de um algoritmo de detecção de pontos de parada.

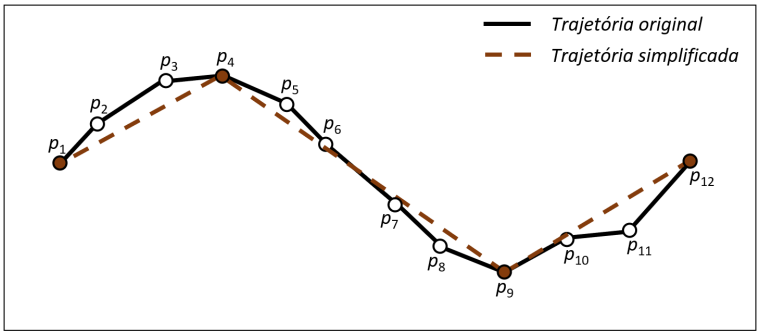


Fonte: (Gomes *et al.*, 2018)

os pontos sem distorcer a sua geometria.

A Figura 4 ilustra uma demostração da simplificação de uma trajetória através da aplicação de uma técnica de compressão de dados de trajetória: ao invés da trajetória ser representada por doze pontos, ela é representada por quatro pontos.

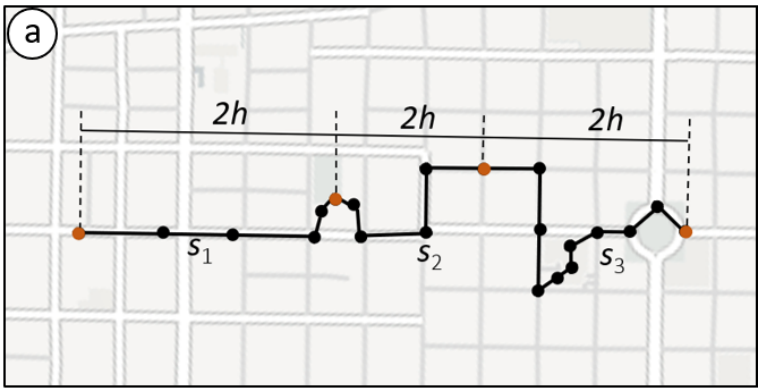
Figura 4 – Demonstração da aplicação de uma técnica de compreensão de dados de trajetórias.



Fonte: (Gomes *et al.*, 2018)

4. **Segmentação**: possui o objetivo de separar e subdividir trajetórias, de tal modo que os segmentos ainda possam manter um comportamento homogêneo. Por exemplo, em relação a velocidade e local, visto que, na maioria dos casos, os dados brutos de trajetória possuem registros por um longo intervalo de tempo, como semanas. Essa grande quantidade de dados podem requerer mais processamento e armazenamento, além de poluir as análises visuais. Geralmente a segmentação é determinada com base em três critérios: tempo, geometria e semântica (HWANG *et al.*, 2018). A Figura 5 apresenta um exemplo de segmentação de trajetória por tempo fixo de duas horas, resultando em três novos segmentos.
5. **Map matching**: possui o objetivo de mapear e adequar os dados de trajetória sob um modelo de representação lógica, como redes urbanas e malha férrea. Os objetos e/ou indivíduos em movimento, limitam-se a transitar dentro dessa representação lógica, portanto,

Figura 5 – Segmentação de uma trajetória com base no intervalo de tempo igual a duas horas.

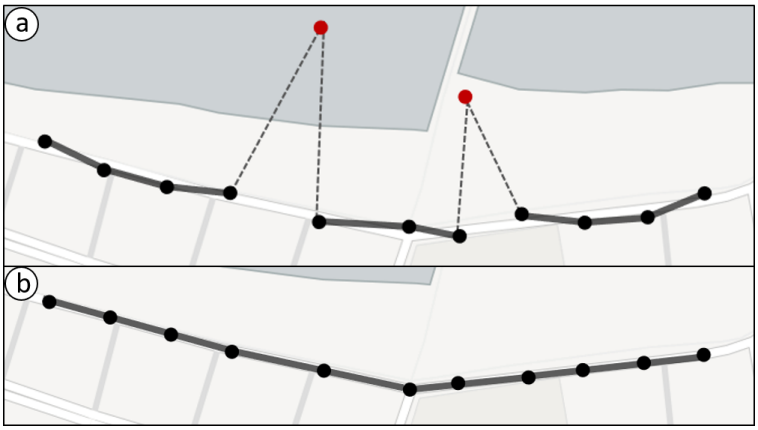


Fonte: adaptado de (Gomes *et al.*, 2018)

os algoritmos de *map-matching* atuam aproximando os pontos de uma trajetória sobre essa representação lógica.

A Figura 6 exemplifica uma aplicação de técnicas de *map matching* para solucionar o problema já apresentado acima e visto em (a), no qual é apresentada uma trajetória com 2 pontos com ruído. Após a execução da técnica é feito o mapeamento dos pontos da trajetória na rede urbana, conforme visto em (b).

Figura 6 – Resultado após a aplicação de uma técnica de *map matching* sobre uma trajetória.



Fonte: adaptado de (Gomes *et al.*, 2018)

2.1.3 Indexação e Recuperação de Trajetórias

Durante o processo de mineração de dados de trajetórias, ocorre a necessidade do frequente acesso a diferentes segmentos e amostras das trajetórias. Frente ao grande volume de dados, esses acessos podem demandar tempo e processamento. Isso torna necessário a adoção de técnicas eficazes para o gerenciamento desses dados, oferecendo a recuperação rápida dos dados.

Em Zheng (2015) são apresentados dois tipos principais de consultas:

- **K-Nearest Neighbors (KNN)**: esse tipo de consulta recupera as K primeiras trajetórias com a distância agregada mínima para alguns pontos ou uma trajetória específica.

  As consultas de ponto preocupam-se em saber se uma trajetória fornece uma boa conexão com os locais/pontos pesquisados, em vez de saber se a trajetória é semelhante à consulta em forma. O número de pontos da consulta também é geralmente muito pequeno e distantes entre si.

  Já a consulta por trajetória é direcionada para encontrar os registros que possuam uma rota ou segmento de trajetória parecida. Essa consulta necessita da definição de uma função que delimite a similaridade e distância entre trajetórias. Além disso, também é necessário o uso de técnicas e algoritmos de processamento eficiente para resolver o problema de pesquisar um grande conjunto de trajetórias.
- **Intervalo de consultas**: recuperam os dados de trajetórias que estão contidos em um espaço ou intervalo. Essa técnica contém três abordagens para consultas de intervalo espaço-temporal.

  A primeira abordagem considera o tempo como a terceira dimensão além das informações que delimitam o espaço geográfico. Com isso, é construída uma *3D-Rtree* com base em trajetórias. O *3D-Rtree* tem boa performance para trajetórias de indexação geradas recentemente, como nas últimas horas. Porém, a perfomance caí quando o período de tempo das trajetórias a serem indexadas dura por um longo período. Isso acontece em detrimento a criação de mais segmentos de trajetórias recém-geradas serem inseridas na 3D-Rtree, acarretando na frequência da sobreposição das representações 3D, e a atualização frequente da estrutura de indexação.

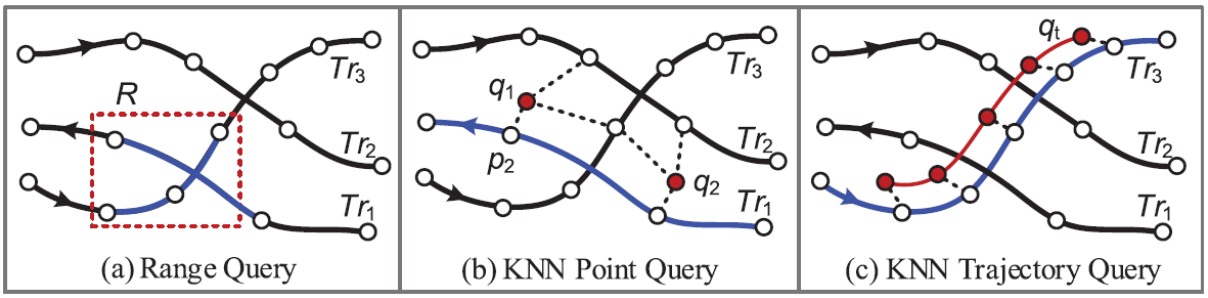
  A segunda abordagem divide um período em vários intervalos de tempo, criando um índice espacial individual como *R-tree* para as trajetórias geradas em cada intervalo. Quando ocorre uma consulta dessa abordagem, o índice primeiro localiza os intervalos de tempo que caem no intervalo temporal e recupera as trajetórias que estão na interseção do intervalo destas consultas.

  A terceira abordagem leva em consideração o espaço geográfico, dividindo-o em grades, as chamadas *grids*, e cria um índice temporal para as trajetórias que caem em cada célula dessa grade. Cada segmento que cai em uma grade é representado por um ponto com as coordenadas iguais ao ponto com horário inicial e o ponto com horário final do segmento.

A Figura 7 mostra as técnicas para melhoria da indexação e recuperação de dados

e segmentos de trajetórias. Em (a) é mostrada a técnica de Intervalo de consulta. Nela são recuperados os pontos e segmentos de trajetórias dentro da região retangular tracejada em vermelho. Já em (b) e (c) são mostradas o uso da técnica de consulta via KNN, sendo a primeira consultas por pontos e a segunta consulta por segmentos de trajetórias similares.

Figura 7 – Demonstração das abordagens utilizadas para indexação e recuperação de trajetórias.



Fonte: Zheng (2015)

2.1.4 Modelos

Muitas aplicações exigem informações instantâneas a partir dos dados de trajetória, como é o caso de aplicações de guia de viagens ou detecção de anomalias de tráfego. Essas informações exigem algoritmos eficazes e muitas vezes são advindas por meio de técnicas de mineração de dados. Essas técnicas podem ser enquadradas em áreas, como classificação, detecção de anomalias ou *outliers*, mineração de padrões e incertezas da trajetória. Abaixo é explorado o conceito de cada uma dessas técnicas.

2.1.4.1 Classificação

Trajetórias e segmentos podem ser classificados de diferentes modos e categorias, como em tipo de atividade, modos de transporte e até mesmo o movimento, através do uso de técnicas e algoritmos de aprendizado supervisionado.

Conforme mostrado em Zheng (2015), em geral, a classificação da trajetória é composta por três etapas principais:

1. Utilização de métodos de segmentação da trajetória para prover segmentos;
2. Extração de características de cada segmento;
3. Criação de modelos para classificar cada segmento.

2.1.4.2 Detecção de anomalias

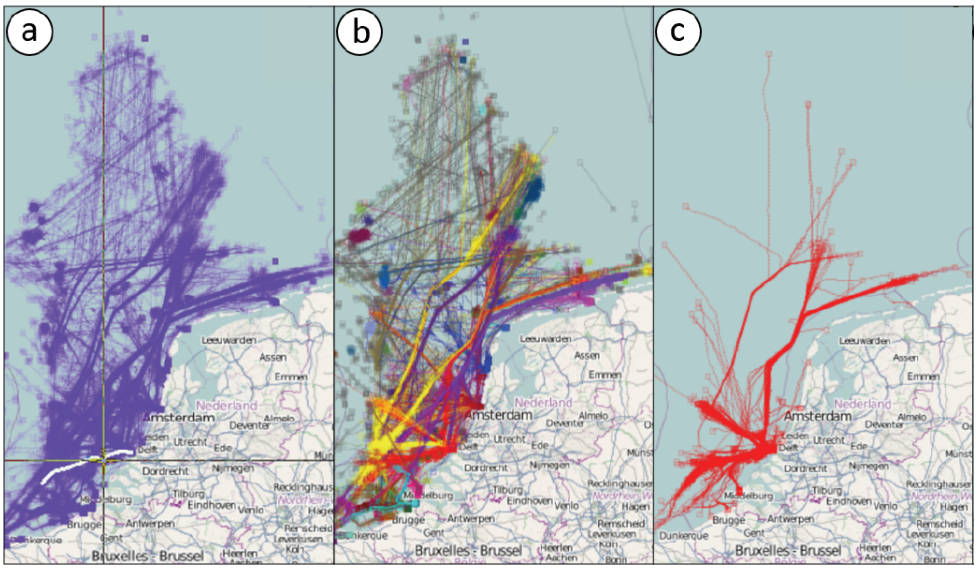
Anomalias, no contexto de dados espaço temporais, consistem em pontos ou até mesmo segmentos de uma trajetória que possuem um comportamento anormal, não seguindo o padrão dos dados que representam uma trajetória, como é o caso representado na Figura 2. Essas anomalias podem estender-se a eventos, desencadeados por algum incidente em determinado momento, como é o caso da velocidade dos veículos diminuírem significativamente diante de um acidente, ou até mesmo desvio de uma rota comum por conta de uma obra ou por estar perdido no caminho. A etapa de detecção de anomalias, para esses dados, consiste na descoberta desses pontos e/ou segmentos que violem um certo padrão de uma trajetória, como picos e quedas inesperados, mudanças de tendência e mudanças de nível. Conforme apresentado em Zheng (2015), essa etapa pode ser realizada por meio de agrupamentos dos pontos e segmentos das trajetórias existentes por meio de padrões. Logo aqueles que não se enquadrarem em algum grupo, possivelmente serão *outliers*.

2.1.4.3 Mineração de padrões

Atualmente há grandes quantidades de dados espaço-temporais que possibilitam a extração de uma gama de informações, como análise e identificação de padrões de mobilidade. Zheng (2015) apresenta quatro categorias principais dentro da mineração de padrões que são apresentados abaixo.

1. **Clusterização** : as técnicas dessa categoria têm como finalidade agrupar trajetórias em busca de algum padrão ou característica que diferenciem grupos.
   Conforme demostrado em Gomes *et al.* (2018), a Figura 8 mostra em (a) a visualização de todas as trajetórias de navios no Mar Norte, representadas por linha. Em (b) são mostradas as trajetórias agrupadas, onde cada grupo/*cluster* é representado por uma cor. Esses grupos são formados com base nos destinos desses navios. Em (c) é mostrado a seleção das trajetórias pertencentes ao *cluster* de cor vermelha.
2. ***Moving Together Patterns*** : essa categoria reúne técnicas que visam detectar indivíduos e/ou objetos que se movem juntos por delimitada janela de tempo, sendo bastante útil para detectar padrões. Esses padrões podem ter diversas aplicações, como ajudar a entender fenômenos de migração e tráfego de espécies.
3. **Padrões Periódicos** : essa categoria reúne técnicas que visam identificar comportamen-

Figura 8 – Resultado da aplicação de técnicas de *clusterização* sobre as trajetórias mostradas em (a) e seleção de um dos *clusters* gerados em (b).



Fonte: Gomes *et al.* (2018)

tos e padrões temporais em trajetórias, como no caso a ida a um dentista durante um intervalo de tempo, ou até mesmo os animais, que migram anualmente de um lugar a outro. Tais comportamentos podem oferecer uma visão concisa acerca de uma longa trajetória, podendo ajudar na compactação de dados da mesma, além de atuar na predição de trajetórias.

4. **Padrões de Frequências Sequenciais** : essa categoria reúne técnicas que possuem a finalidade de padrões sequenciais, onde objetos seguem por uma sequência comum de lugares, não necessariamente consecutiva, com uma janela de tempo semelhante, a partir de uma ou múltiplas trajetórias. Essas técnicas são amplamente utilizados para recomendações de viagem e outras finalidades conforme abordado em (ZHENG, 2015).

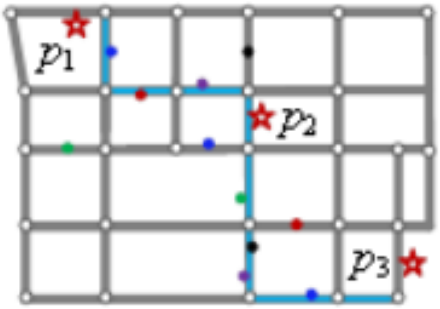
2.2 Incerteza em trajetórias

Indivíduos e objetos realizam movimentos contínuos em determinados locais em que trafegam. Porém, devido à limitação de algumas aplicações ou por questão de economia de energia dos sensores, os dados que representam essa trajetória são enviados periodicamente. Desse movo, a localização desse indivíduo e/ou objeto seja incerta, ou até mesmo desconhecida, entre dois pontos que descrevem sua trajetória.

A Figura 9 apresenta uma demonstração de uma trajetória formada por três pontos. Nela é perceptível que entre dois pontos é possível tomar caminhos diferentes, por exemplo:

partindo do ponto p_1 é possível chegar ao ponto p_2 de diferentes formas. Este contexto ocasiona a incerteza dos caminhos tomados entre pontos de uma trajetória.

Figura 9 – Demonstração de uma trajetória formada por três pontos, p_1, p_2 e p_3.



Fonte: adaptado de (ZHENG, 2015)

Nessa linha de pesquisa, há duas sub-áreas, descritas a seguir:

- **Redução de incertezas**: consiste em desenvolver técnicas que visam diminuir a incerteza da trajetória entre dois pontos. Em Wei *et al.* (2012), por exemplo, é descrita uma técnica que faz uso de trajetórias incertas para construção de rotas populares.
- **Preservação de privacidade**: consiste em desenvolver técnicas que visam ampliar a incerteza de trajetórias sem afetar a qualidade de um serviço ou o uso desses dados. Isso é feito visando proteger um usuário do vazamento de privacidade causado pela divulgação de suas trajetórias (ZHENG, 2015).

2.3 Bibliotecas para manipulação de dados para Python

Para trabalhar com dados de trajetória é necessário utilizar estruturas de dados adequadas para otimizar o processamento dos mesmos. Nesta seção, são descritas as duas bibliotecas utilizadas neste trabalho, que são amplamente utilizadas pela comunidade do Python, para representação e manipulação de dados.

2.3.1 Pandas

Pandas, Mckinney (2011), é uma biblioteca de código aberto para Python, patrocinada pela NumFOCUS [3]. Ela fornece estruturas de dados rápidas, flexíveis e expressivas, projetadas para facilitar o trabalho com dados. O objetivo é ser o bloco de construção de alto

[3] https://numfocus.org/

nível fundamental para a análise prática dos dados.

É construída sobre o NumPy, van der Walt *et al.* (2011), com uma proposta de integrar-se bem em um ambiente de computação científica com muitas outras bibliotecas de terceiros. Trata e lida com diversos tipos e fontes de dados, como tabulares com colunas com diferentes tipos, séries temporais ordenados ou não e matrizes. As duas estruturas de dados primárias dos Pandas são:

- **Series**: são *arrays* unidimensional, que possui um índice que dá rótulos a cada elemento da lista.
- **DataFrame**: é uma estrutura de dados flexíveis com duas dimensões onde linhas podem ter colunas de tipos diferentes, como listas, vetores, ou até mesmo outros DataFrames. Cada coluna é uma Serie.

2.3.2 Dask

Dask, Rocklin (2015), é uma biblioteca disponível para Python, flexível para computação paralela de código aberto. Faz uso de *Application Programming Interface* (API) Python e estruturas de dados existentes para facilitar a alternância entre o Numpy, o Pandas e o Scikit-learn, (PEDREGOSA *et al.*, 2011), para seus equivalentes do Dask. Possui características importantes como familiaridade de sintaxe, flexível, rápido e escalável.

Essa biblioteca é composta de duas partes: tarefas otimizadas para computação e para cargas de trabalho computacionais, e coleções para tratar *Big Data*, como matrizes paralelas, quadros de dados e ambientes maiores que a memória *Random Access Memory* (RAM) ou distribuídos.

2.4 Visualização dos dados

Em paralelo ao crescimento da aquisição de dados espaço temporais, há também a evolução dos sistemas computacionais, principalmente no âmbito de armazenamento e processamento. Isso possibilita que esses dados possam ser processados e utilizados em inúmeros tipos de aplicações.

Embora sejam utilizadas técnicas de mineração desses dados, automatizar totalmente esse processo pode não ser eficiente. Por isso, realizar visualizações exploratórias permitem que os indivíduos possam incorporar o seu conhecimento do domínio, o que permite criar associ-

ações, detectar e descrever detalhes e padrões ainda não revelados nos dados (ANDRIENKO; ANDRIENKO, 2013).

Para visualizar dados de trajetória são necessárias técnicas de visualização adequadas. Essas técnicas exploram canais visuais como cor, tamanho, forma, orientação aplicadas a diferentes marcas como pontos, linhas, áreas, volumes e superfícies para representar estes dados, transformando-os em representações visuais apropriadas. A vantagem do uso de visualizações é a incorporação de capacidades humanas em uma interface visual intuitiva, combinando inteligência de máquina com inteligência humana. Essas visualizações podem ser combinadas com procedimentos de processamento dos dados, visando diminuir, limpar e filtrar os dados.

A aplicação de técnicas de visualização de dados nos grandes volumes de dados de trajetórias podem facilitar a compreensão do comportamento de objetos em movimento, como veículos e descoberta de tráfego, social, geoespacial e até padrões econômicos (CHEN *et al.*, 2015).

Em Morshed *et al.* (2019) é descrita uma ferramenta de visualização de dados de trajetórias de crime. Esse tipo de trajetória possui características distintas de outros tipos de trajetórias, o que fortalece o uso de análises visuais sobre esses dados. Já Chen *et al.* (2015) destaca e explora as classificações dos tipos de visualização frente aos tipos de tarefas a serem realizadas na análise de tráfego.

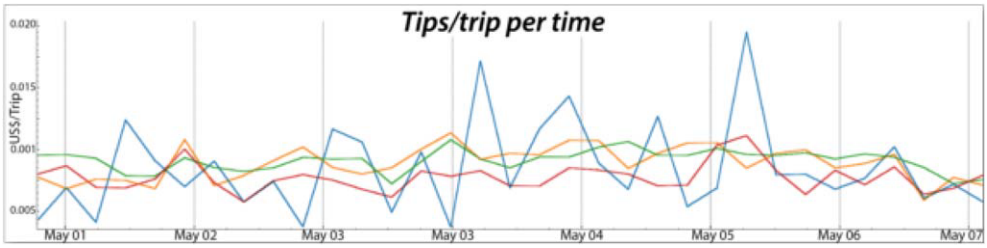
Abaixo serão discutidas algumas formas de visualização e análise de dados de trajetórias que nos permitem extrair informações ou até mesmo visualizar de forma mais simples o grande volume de dados.

2.4.1 Gráficos de Linhas

A visualização orientada ao tempo enfatiza a exibição de padrões, periodicidade, tendências e anormalidade dos dados de trajetória. Quando a representação do tempo é linear, é considerado um intervalo entre um ponto inicial e final. Nesse tipo de representação é amplamente utilizado gráfico de linhas, onde o eixo X delimita o tempo e o Y outra característica do dado, como os picos de velocidade de uma trajetória.

A Figura 10 mostra uma visualização dos dados de gorjetas das viagens de táxis fazendo uso do gráfico de linhas para demonstrar a quantidade de gorjetas obtidas pelas viagens de táxi no período de tempo de 1 a 7 de maio de 2011. Cada linha representa a gorjeta por viagem em uma região.

Figura 10 – Gráfico de linhas seguindo a representação de tempo linear.



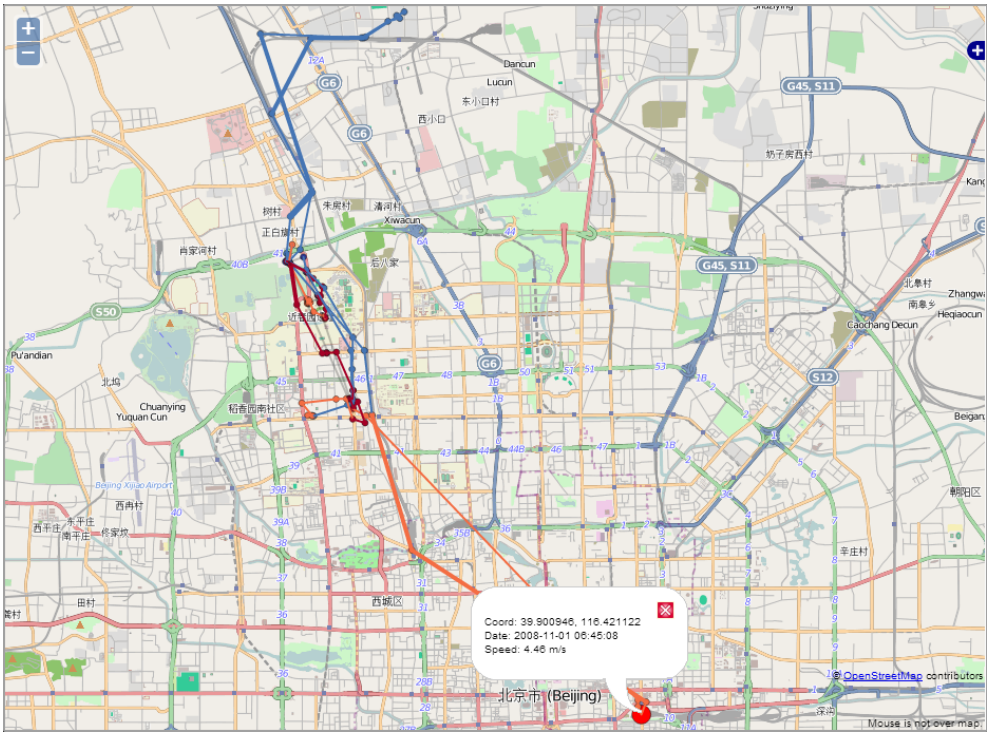
Fonte: Chen *et al.* (2015)

2.4.2 Mapas estáticos

Os mapas estáticos bidimensionais são abordagens bastante comuns para representar informações de localização geográfica, como a análise de trajetórias de mobilidade ou fluxo de tráfego em uma rede distribuída. Convencionalmente, uma trajetória é representada por uma linha ou uma curva, podendo explorar combinações de variáveis, como cor, tamanho e direção, em relação às suas propriedades.

A Figura 11 exemplifica a utilização de diversos recursos de visualizações: pontos para representação de eventos, linhas para representação de segmentos de trajetória, cores para diferenciar e possivelmente classificar os tipos de trajetória.

Figura 11 – Exploração do uso de recursos visuais para visualização em mapas estáticos.



Fonte: Gonçalves *et al.* (2013)

2.4.3 Visualizações com mapas animados

Neste tipo de visualização os dados são divididos entre vários mapas, cada um representando o estado de um fenômeno e/ou intervalo de tempo. Este tipo de visualização pode ajudar a revelar padrões temporais que não são evidentes com análises estáticas. Além disso, diferentemente das abordagens estáticas, elas têm uma dimensão adicional que pode ser usado para apresentar informações. Entretanto, a ocultação das camadas periodicamente, pode acarretar a limitação perceptivas. (GONçALVES *et al.*, 2013).

Essas visualizações também fornecem interatividade para o usuário final, permitindo a visualização das camadas desejadas, amenizando o problema relatado acima. A Figura 12 apresenta um mapa dinâmico que representa o movimento de um ponto ao longo de quatro visualizações.

Figura 12 – Visualização do movimento de um ponto através do uso de Mapas dinâmicos.

Fonte: Gonçalves *et al.* (2013)

2.4.4 Cluster

Quando se tem um conjunto de dados com uma quantidade significativa de pontos de trajetórias, torna-se difícil realizar análises sobre uma visualização, seguindo abordagens tradicionais, que contenha todos esses pontos. Em muitos casos, pontos ficam sobrepostos a outros. A Figura 16c demonstra esse problema: há uma grande quantidade de pontos de trajetórias espalhados sobre o mapa, acarretando em poluição visual e prejudicando a análise.

Visualizações com *clusters* em mapas ajudam bastante em análises, visto que um grupo de pontos que em determinado momento se sobrepunham, tornam-se um só. Essa técnica permite obter uma visão geral rápida de seus conjuntos de *clusters*. Cada *cluster* é relativamente dimensionado ou rotulado com o número de pontos que possuem. Esse tipo de visualização é ideal em mapas interativos, onde o usuário pode averiguar quais pontos compõem cada *cluster*.

2.4.5 Mapa de Calor

Mapa de Calor é um método advindo de uma generalização de um gráfico de dispersão, que gera uma visualização a partir de uma matriz de células, onde cada célula é colorida de forma gradiente com base em valores ou função dos dados, sendo bastante útil quando é necessário gerar visualizações para grande volume de dados, fornecendo uma visão geral dos maiores e menores valores dos dados (METSALU; VILO, 2015).

Esse método é usado em diferentes contextos, por exemplo: em Matejka *et al.* (2013) é apresentado um sistema que coleta e visualiza quais as partes mais utilizadas de uma aplicação por meio de mapas de calor; em Ha *et al.* (2009) é mostrado o uso dessa técnica para visualização de atividade de enzimas, de dados geográficos e de trajetórias. A estimativa de densidade de *kernel*, Kernel Density Estimation (KDE), é um algoritmo comumente usado para gerar um mapa de calor.

A Figura 13 demonstra o uso da técnica Mapa de Calor para visualização das rotas quentes de uma cidade. As regiões avermelhadas concentram um grande volume de tráfego, enquanto as regiões azuis indicam o baixo volume.

Figura 13 – Visualização das rotas quentes em um cidade utilizando a técnica Mapa de Calor.

Fonte: Chen *et al.* (2015)

2.4.6 Choropleth

Choropleth é uma técnica que consiste em visualizar zonas e áreas geográficas sob um mapa, onde estas são coloridas em relação a uma faixa de valor de alguma variável nos

dados.

Essa técnica permite analisar o comportamento de uma variável ao longo de uma área. São bastantes utilizados para visualizar intuitivamente agrupamentos ou concentração de dados geográficos. Porém, esse tipo de visualização pode gerar interpretações equivocadas em situações em que a variável analisada estiver vinculada ao tamanho da área, fazendo com que o tamanho de uma região influencie diretamente na visualização, ou até mesmo obstrua pequenas regiões.

2.4.7 Ferramentas para visualização em Python

No Python, existem algumas bibliotecas e *plugins* que possibilitam a construção de gráficos e visualizações dos dados. O PyMove faz uso de algumas bibliotecas e *plugins*, tais como:

- **Matplotlib** (HUNTER, 2007): é uma biblioteca de geração e visualização com duas dimensões do Python. Ela facilita a geração de visualizações com apenas algumas linhas de código, oferecendo suporte para gráficos como histogramas, espectros de potência, gráficos de barras, gráficos de erros e gráficos de dispersão.
- **Folium**[4]: é uma biblioteca do Python que facilita a visualização de dados que foram manipulados no Python em um mapa de Leaflet[5], biblioteca JavaScript de código aberto para mapas interativos. Ele permite a ligação de dados a um mapa para visualizações, bem como a transmissão de visualizações vetoriais e *HyperText Markup Language* (HTML) avançadas como marcadores no mapa.

  Além disso, o Folium possui integração com vários conjuntos de *tiles*, que são blocos de imagens usados para criação de um mapa e possui *plugins* que permitem adaptar e gerar outros tipos de visualizações.

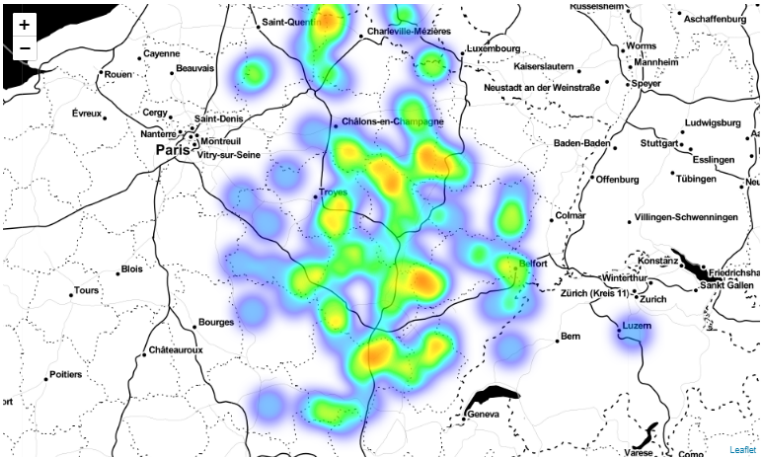
  A Figura 14 apresenta uma visualização utilizando a técnica Mapa de Calor, Seção 2.4.5, gerada por meio de *plugin Heat Map*. A Figura 15 mostra uma visualização que faz o uso do *plugin*. Nela há a demonstração a taxa de desemprego em alguns estados dos Estados Unidos. No Folium, também é possível gerar as visualizações de *clusters* através do *plugin MarkerCluster*, onde são inicialmente criados *clusters* que, de acordo com o nível de zoom, podem agrupar outros *clusters* ou subdividir em novos, até chegar aos seus pontos

[4] Documentação Folium: https://python-visualization.github.io/folium/
[5] https://leafletjs.com/

originais. Há uma adaptação chamada *FastMarkerCluster*, que é bastante útil levando em consideração um grande número de pontos, e requerendo uma visualização com um tempo de resposta mais rápido.

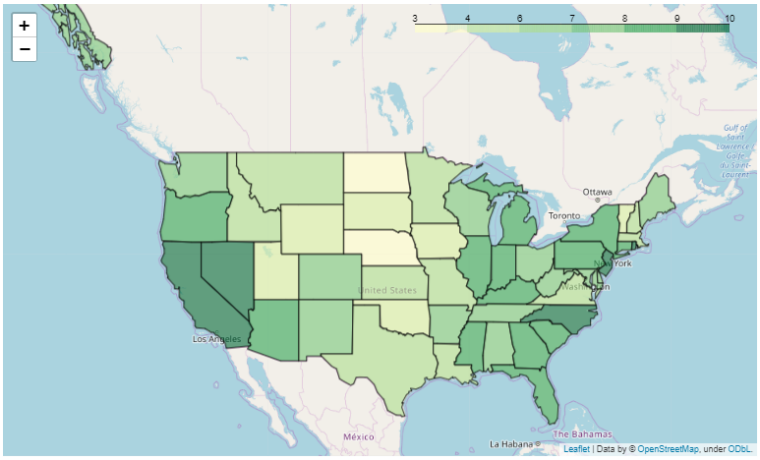
A Figura 16a demonstra de como a visualização fica poluída quando se há um grande conjunto de dados. A Figura 16b apresenta uma nova visualização utilizando o *plugin MarkerCluster* ou *FastMarkerCluster* com a mesma quantidade de pontos utilizada em 16a. As Figuras 16c e 16d mostram a diferença entre o uso dos dois *plugins* citados anteriormente, que se caracteriza no uso de um marcador ou um círculo, respectivamente.

Figura 14 – Visualização utilizando a técnica Mapa de Calor utilizando o Folium.



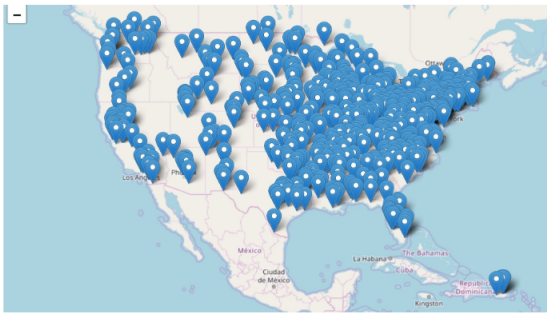
Fonte: Documentação Folium

Figura 15 – Exemplo de visualização utilizando a técnica *Choropleth* por meio do Folium.

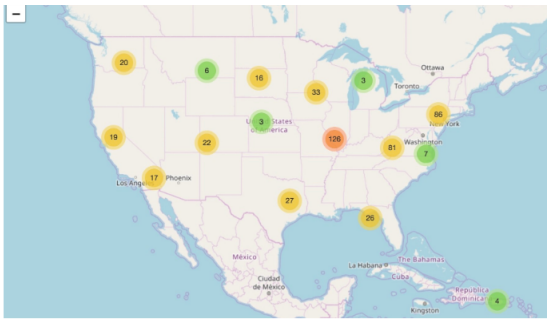


Fonte: Documentação Folium

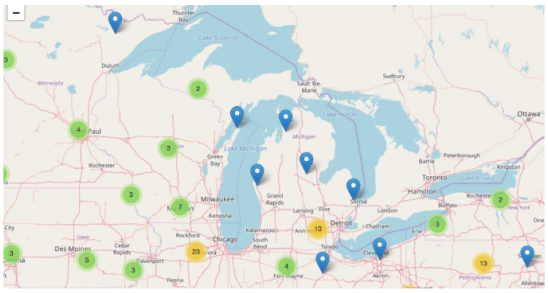
Figura 16 – Comparação de visualizações geradas pelo Folium quando se há muitos dados utilizando apenas marcadores e com a técnica de *cluster*.



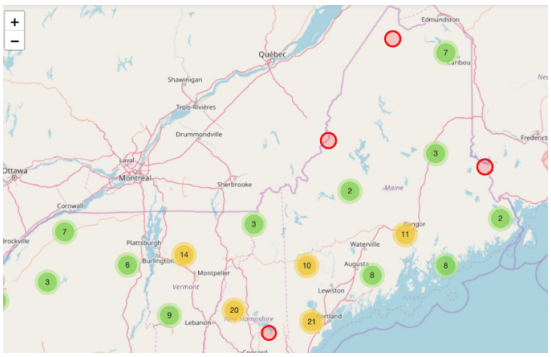
(a) Visualização de todos os pontos sob um mapa gerado pelo Folium.



(b) Visualização da mesma quantidade de pontos utilizando o *plugin* de cluster do Folium.



(c) Visualização utilizando *MarkerCluster*



(d) Visualização utilizando *FasterMarkerCluster*

Fonte: Documentação Folium

2.5 Padrões de projetos

O uso de Padrões de Projeto é uma boa prática de desenvolvimento, por ser uma solução geral para um problema em determinado contexto. Isso facilita outros usuários entenderem o objetivo e problema abordado. Além disso, otimizam o processo de trabalho por possibilitarem a reutilização de soluções para determinados problemas.

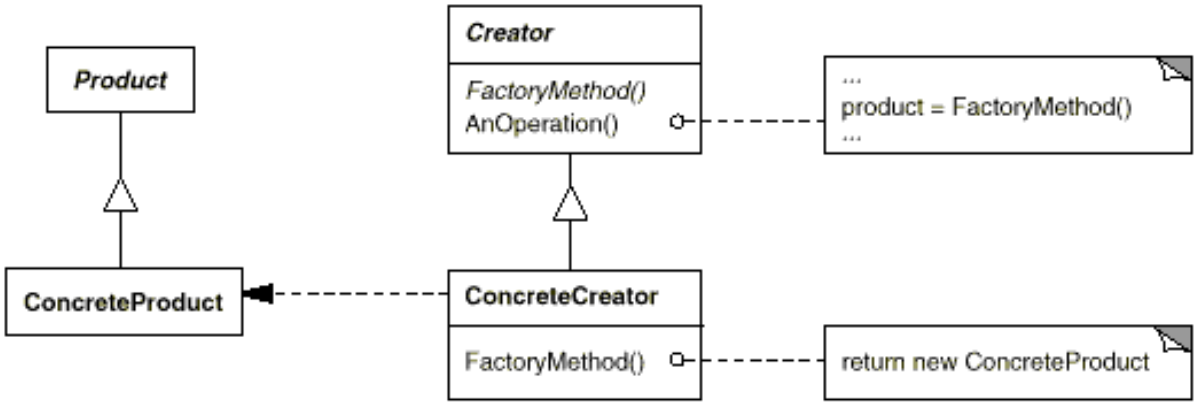
Neste trabalho é adotado o padrão de projeto *Factory Method*. Este padrão permite às classes definirem um método separado para criação dos objetos, no qual as subclasses possam sobrescrever para especificar o "tipo derivado"do produto que vai ser criado.

A Figura 17 apresenta um diagrama de classes desse Padrão de Projeto. Gamma *et al.* (1995) definem quatro participantes desse padrão:

- ***Product***: interface que define as características e métodos para os objetos criados pelo *Factory Method*.
- ***ConcreteProduct***: classe concreta que estende e implementa os métodos de um *Product*.
- ***Creator***: interface que possui o *factory method* que retorna o objeto da classe *Product*.

- ***ConcreteCreator***: classe concreta que estende a classe *Creator* e de fato implementa o método de fabricação.

Figura 17 – Padrão *Factory Method*.



Fonte: (GAMMA *et al.*, 1995)

2.6 Boas práticas de programação

No processo de desenvolvimento de códigos é importante seguir diretrizes e convenções do ambiente e linguagem que estão sendo utilizadas. Essas convenções geralmente delimitam boas práticas de programação, que ajudam a tornar o código mais limpo e legível. Além disso, há maiores chances do código produzir menos erros e identificá-los mais rapidamente.

Em Python existem as *Python Enhancement Proposal* (PEP) que proveem informações sobre a comunidade Python e guias de estilo para programação. Dentro desse contexto de convenções de código, entram as ferramentas de análise estática. Elas ajudam os desenvolvedores a identificarem problemas antecipadamente, tal como a sua solução.

2.6.1 PEP8

A PEP8 é uma versão da PEP, criada por Guido van Rossum, Barry Varsóvia e Nick Coghlan, em 2001. Desde então, evolui à medida que convenções são adicionadas e ficam depreciadas. Este documento fornece convenções de codificação para o código Python.

2.6.2 Análise estática

A análise estática é uma técnica utilizada na etapa de Verificação de um *software*. Consiste em realizar a depuração do código de um programa de computador sem executa-lo. A técnica proporciona:

- Compreensão da estrutura do código;
- Verificação do estilo e boas práticas do código-fonte;
- Adequação do código-fonte às normas estabelecidas pela linguagens ou adotadas em um determinado domínio;
- Encontrar e solucionar erros e problemas no código-fonte.

Ferramentas automatizadas podem ajudar os desenvolvedores na realização de uma análise estática. Para este trabalho foi adotado a ferramenta o Pylint[6], que é uma ferramenta de análise estática que verifica erros de tipo e *code smells*, oferece recomendações e sugestões de boas práticas de programação e detalhes sobre a complexidade em códigos que utilizam a linguagem de programação Python.

Após a análise também são apresentadas estatísticas sobre o número de avisos e erros encontrados, uma classificação geral com base no número e gravidade dos erros, além das mensagens de erros e avisos, classificadas em várias categorias.

[6] https://pylint.readthedocs.io/en/latest/

3 TRABALHOS RELACIONADOS

Este capítulo apresenta os trabalhos mais relevantes relacionados a ferramentas que oferecem manipulação e técnicas aplicadas a trajetórias, com enfoque no suporte para visualização dos dados espaço-temporais. No final, na Seção 3.4, são apresentadas conclusões finais e o quadro comparativo entre os trabalhos apresentados e este trabalho.

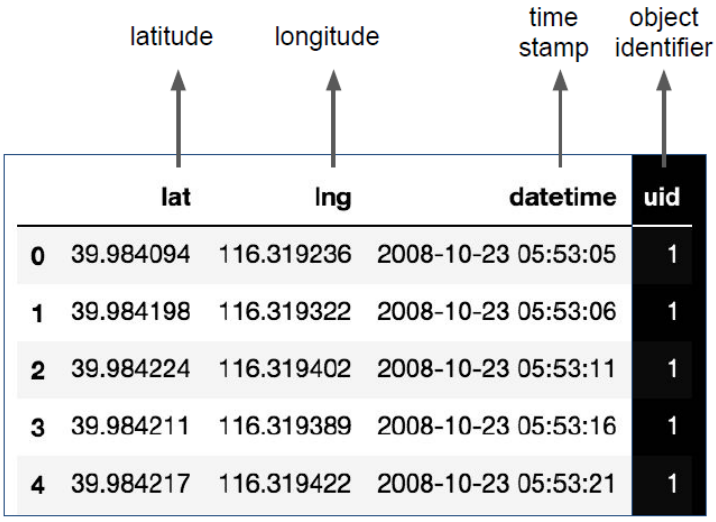
3.1 scikit-mobility

Pappalardo *et al.* (2019) apresentam a biblioteca scikit-mobility, disponível para a linguagem Python, que tem os objetivos e diretrizes idênticas ao que o PyMove propõe: fornecer um *software*/ferramenta que contém um módulo para visualização para auxiliar outros pesquisadores a trabalharem com dados de trajetória, trazendo técnicas disponíveis na literatura.

Essa biblioteca fornece estruturas de dados, que usam como base o *DataFrame* do Pandas, descrito na Seção 2.3.1. Essas estruturas de dados oferecem suporte para representação de um conjunto de trajetórias, no caso o *TrajDataFrame*, e uma representação para matrizes de origem e destino, também conhecidas como matriz de fluxo, no caso o *FlowDataFrame*.

A Figura 18 mostra como é a estrutura *TrajDataFrame*. Essa estrutura tem, obrigatoriamente, dados referentes a latitude, longitude e *time stamp*. Também pode ter colunas opcionais para representar uma trajetória e/ou um usuário, no caso o campo *uid* presente na Figura. Já na estrutura *FlowDataFrame*, cada linha representa um fluxo de objetos entre dois locais, descritos por três colunas obrigatórias: origem, destino, e fluxo.

Figura 18 – Estrutura da *TrajDataFrame*.



Fonte: (PAPPALARDO *et al.*, 2019)

O scikit-mobility oferece módulos para a etapa de pré-processamento, citadas na Seção 2.1.2; técnicas para geração de dados sintéticos; análise dos padrões estatísticos de trajetórias; técnicas para avaliação do risco de privacidade relacionado à análise de conjuntos de dados de mobilidade.

A biblioteca também propõe técnicas para geração de visualizações, que possam ajudar na fase de análise exploratória de dados de conjuntos de dados de mobilidade. Essas visualizações são geradas utilizando o Folium, descrito na Seção 2.4.7. As visualizações que podem ser geradas são:

- Linhas conectando todos os pontos da trajetória. Devido a possibilidade de haver quantidades massantes de dados, oferecem limites de dados a serem utilizados para visualização, por meio da delimitação de número máximo de pontos e o número máximo de usuários. Um exemplo dessa visualização pode ser vista na Figura 19a;
- Localização de pontos de paradas, por meio de marcadores. Novamente oferece recursos para limitação de pontos a serem utilizados na visualização. Um exemplo dessa visualização pode ser vista na Figura 19b;
- Sequência de locais visitados ao longo do tempo por um indivíduo e/ou objeto;
- GeoDataFrame[1], é um DataFrame do Pandas, descrito na Seção 2.3.1, que possui uma coluna com uma geometria, associado a estrutura do *FlowDataFrame*, presente na Figura 19c;
- Fluxos de movimentos, representados na estrutura *FlowDataFrame*, presente na Figura 19d.

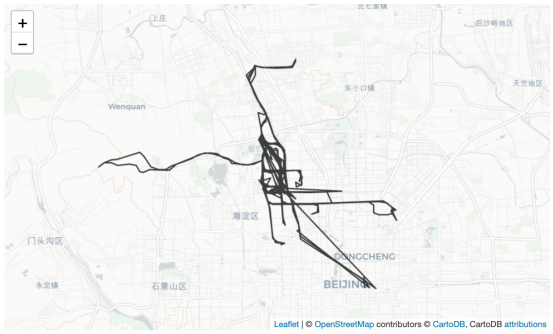
Ainda em Pappalardo *et al.* (2019), é destacado que visa-se desenvolver mais módulos para análise desses dados, incluindo técnicas de predição e map-matching; e melhorar o desempenho da biblioteca em âmbito de processamento, permitindo escalabilidade, por meio de uso de outras estruturas, como o Dask, descrito na Seção 2.3.2.

3.2 CloudTP: A Cloud-based Flexible Trajectory Preprocessing Framework

Em Ruan *et al.* (2018), é proposta a ferramenta online *CloudTP*, que oferece técnicas de pré-processamento e visualização de dados. Faz uso da computação paralela, através do uso do Spark, apresentado em (ZAHARIA *et al.*, 2010), e a nuvem para armazenamento. O uso dessas abordagens permitem melhorar a eficiência do processamento dos registros de trajetórias

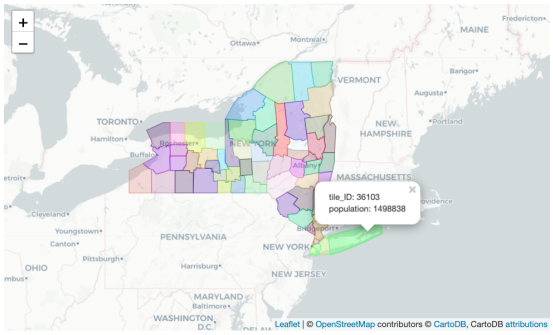
[1] http://geopandas.org/reference/geopandas.GeoDataFrame.html

Figura 19 – Visualizações geradas pela biblioteca Scikit-Mobility.

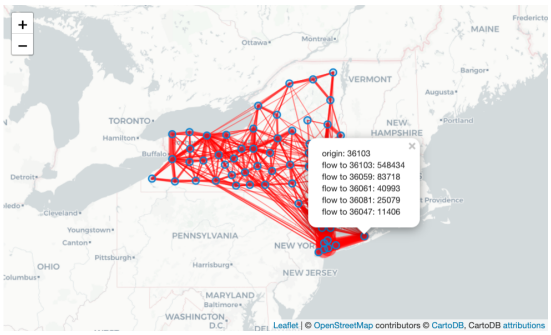


(a) Visualização de todas as trajetórias.

(b) Visualização dos pontos de parada.



(c) Visualização do GeoDataFrame.



(d) Visualização dos fluxos de movimentos.

Fonte: (PAPPALARDO *et al.*, 2019)

em larga escala e acelerar a execução de tarefas.

É uma ferramenta que se mostra flexível diante dos tipos de viagem que os dados podem trazer, além de dispor de dois níveis de acesso a ferramenta:

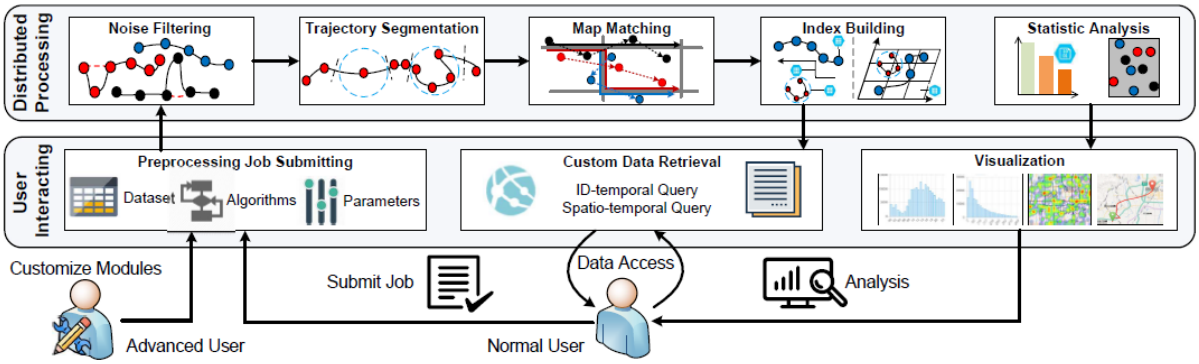
- Usuários normais: usam a ferramenta apenas para o processamento dos dados, seguindo o padrão da ferramenta, e recuperam os resultados.
- Usuários avançados: podem personalizar o processamento dos dados, não seguindo obrigatoriamente o padrão disposto pela ferramenta, para mostrar a capacidade e flexibilidade do CloudTP.

A Figura 20 mostra a estrutura da ferramenta: há uma camada para interação do usuário para fornecimento dos dados e solicitações de trabalho, e recuperação dos dados procesados e análises visuais; há uma camada de processamento distribuído, que envolve as técnicas de pré-processamento citadas acima e geração de análises visuais e estatística.

A Figura 21 apresenta as interfaces da ferramenta. A Figura 21a mostra o processo de *upload* dos dados de trajetória para o Azure Blob[2], possibilitando o usuário, como descrito em (RUAN *et al.*, 2018), especificar as redes de estradas de uma cidade, ajustar parâmetros e selecionar o modo de viagem. Cada trabalho receberá um token exclusivo. A Figura 21b mostra

[2] https://azure.microsoft.com/en-us/services/storage/blobs/

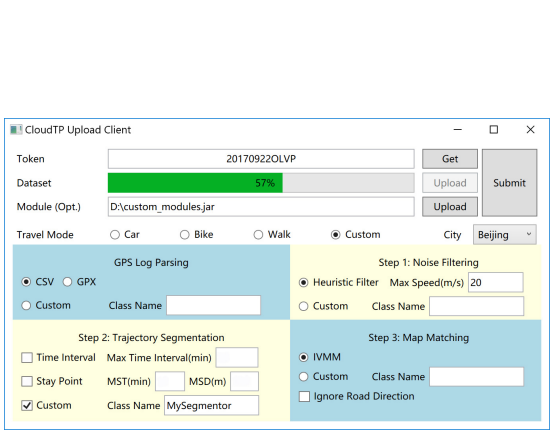
Figura 20 – Estrutura em alto nível da ferramenta *CloudTP*.



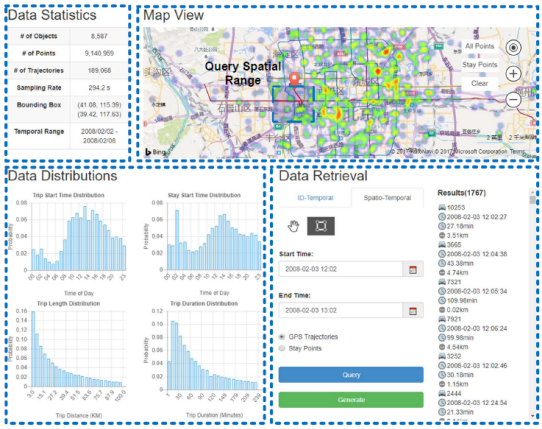
Fonte: (RUAN *et al.*, 2018)

os resultados obtidos, desde análises estatísticas e visuais. Nela é possível notar o uso de técnicas como *Heat Map*, descrito na Seção 2.4.5, para visualização dos dados.

Figura 21 – Interfaces gráficas da ferramenta *CloudTP*.



(a) Interface para *upload* de arquivos.



(b) Interface com resultado do processamento.

Fonte: (RUAN *et al.*, 2018)

3.3 PySAL

Em Rey e Anselin (2007) é proposta uma biblioteca para Python, *open source*, voltada para aplicação de técnicas de ciência de dados geoespaciais, supontando desenvolvimento de aplicativos de alto nível para análise espacial, variando desde detecção de clusters espaciais, anomalias, regressão espacial e modelagem estatística, a econometria espacial.

Essa biblioteca conta com os módulos *explore*, que possui módulos para realizar análises exploratórias de dados espaciais e espaço-temporais, incluindo testes estatísticos em pontos e redes poligonais; *viz*, que inclui métodos para visualizar padrões em dados espaciais para detectar *clusters*, anomalias e *hot-spots*; *model*, que modela relações espaciais em dados

com uma variedade de modelos lineares e não lineares; e, o módulo *lib* que possui métodos para tratar de uma variedade de problemas de geometria computacional.

3.4 Análise entre os trabalhos

O Quadro 1 apresenta a comparação deste trabalho e os trabalhos desenvolvidos em Pappalardo *et al.* (2019), Rey e Anselin (2007) e Ruan *et al.* (2018). Neste são avaliados os seguintes pontos:

- **Plataforma**: descreve o escopo do trabalho avaliado e a linguagem de programação em que foi implementado;
- **Propriedades**: avalia se o trabalho possui as características de **flexibilidade**, referente ao usuário poder trocar entre estruturas de dados, modificar módulos; **escalabilidade**, referente a comportar o trabalho com *Big Data*; **extensibilidade**, referente ao produto comportar facilmente a adição de novas estruturas de dados e/ou novas implementações; e, por fim, **paralelismo**, que refere-se a aplicação suportar computação paralela ou não.
- **Módulos**: descreve os módulos presentes na arquitetura da aplicação;
- **Técnicas de visualizações**: descreve as técnicas utilizadas no trabalho, seguindo as definições abordadas na Seção 2.4;
- **Disponibilidade**: indica se o produto gerado no trabalho está disponível para uso, onde "disponível"indica que a aplicação ou ferramenta está disponível para download e/ou uso. Já "não se aplica"refere-se que, até a elaboração deste trabalho, a referência para uso da ferramenta está indisponível.

O trabalho aqui proposto se assemelha bastante com o produzido em Pappalardo *et al.* (2019), pois ambos sugerem estruturas de dados, que utiliza o DataFrame do Pandas, Seção 2.3.1, além de uma arquitetura que comporta as operações mostradas na Figura 1. Esse trabalho também oferece um módulo de visualização que faz uso, principalmente, do Folium para gerar as representações gráficas. O scikit-mobility provê mais operações do que as disponíveis no PyMove. Em contrapartida, não provê as características de flexibilidade, extensibilidade, escalabilidade e nem paralelismo. A biblioteca proposta em Rey e Anselin (2007) segue as mesmas características citadas acima, porém com módulos mais acoplados e defasados, e com mais visualizações disponíveis.

Já a ferramenta desenvolvida em Ruan *et al.* (2018) possui um escopo limitada de visualizações e operações, porém oferece flexibilidade ao usuário final para adaptar as operações

Quadro 1 – Comparativo das características dos Trabalhos.

Trabalhos	**Plataforma**	**Propriedades**	**Módulos**	**Técnicas de Visualizações**	**Disponibilidade**
scikit-mobility	Biblioteca em Python	Não-flexível; não-escalável; não-extensível; não-paralelo.	Estrutura de dados; pré-processamento; modelos; privacidade de dados; e visualização.	Mapa estáticos; Choropleth.	Disponível
CloudTP	Ferramenta Online em Java	Flexível; escalável; não-extensível; paralelo.	Pré-processamento; e visualização.	Heat Map; Histograma; Mapa estáticos.	Não se aplica
PySAL	Biblioteca em Python	Não-flexível; não-escalável; não-extensível; não-paralelo.	*explore;* *viz;* *model;* *e lib.*	Choropleth; Mapas estáticos; Gráfico de linhas; Histogramas.	Disponível
Este trabalho	Biblioteca em Python	Flexível; possivelmente escalável; extensível; possivelmente paralelo.	Estrutura de dados, pré-processamento, modelos, privacidade de dados e visualização.	Gráfico de Linhas; Histogramas; Heat Map; Cluster; Mapas estáticos.	Disponível

Fonte: elaborado pelo autor.

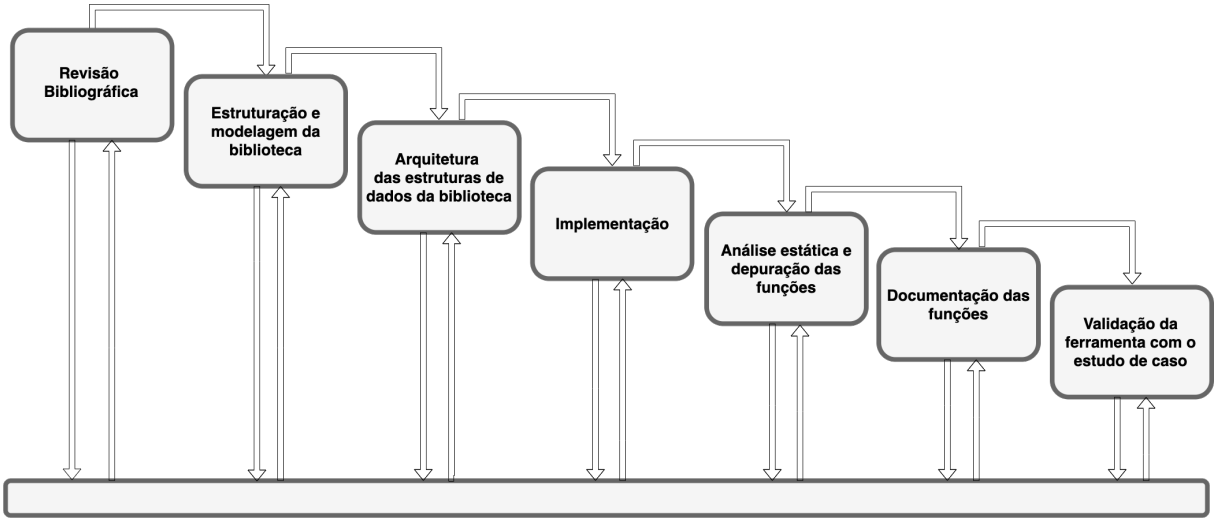
disponíveis. Além disso suporta computação paralela e escalabilidade. Em contrapartida, a ferramenta não está disponível.

Com isso, este trabalho busca oferecer uma modelagem semelhante ao proposto em Pappalardo *et al.* (2019), porém de forma que torne-se facilmente extensível e flexível ao usuário final; abordar técnicas de visualizações dispostas no módulo *viz* do trabalho proposto em Rey e Anselin (2007); e reunir as características e benefícios trazidos pela computação paralela e escalabilidade, demonstrados em Ruan *et al.* (2018).

4 MODELAGEM E DESENVOLVIMENTO DO PYMOVE PARA ANÁLISE E VISUALIZAÇÃO DE DADOS

Este capítulo apresenta as descrições das etapas e procedimentos realizados para o desenvolvimento deste trabalho. A Figura 22 demonstra as etapas e o fluxo de trabalho seguido. Essas estapas estão descritas nas Seções de Modelagem da biblioteca, 4.1, Desenvolvimento, 4.2.1 e Validação, 4.3.

Figura 22 – Visão geral da Metodologia



Fonte: elaborada pela autora

Antes da execução das demais etapas deste trabalho, foi realizado a análise da literatura e revisão bibliográfica acerca da área de trajetórias, seus principais conceitos e técnicas. Esse estudo possibilitou o entendimento do que é e como é representada uma trajetória, das informações que podem ser extraídas e as aplicações existentes a partir desses dados, presentes na Seção 2.1; as técnicas de pré-processamento, descritas na Seção 2.1.2; os modelos que podem ser aplicados nos dados de trajetória, descritos na Seção 2.1.4; por fim, também foi possível achar os trabalhos relacionados descritos na Capítulo 3.

Nas pesquisas voltadas para as formas de gerar visualizações para dados espaço-temporais, foram encontradas algumas bibliotecas que fornecem um forte arcabouço, dentre elas as utilizadas nesse trabalho estão descritas nas Seções 2.4.7 e 2.4.7.

Os principais trabalhos que forneceram essa base teórica para o desenvolvimento desse trabalho foram:

- **Zheng (2015)**: esse trabalho é um *Survey* que engloba as linhas de pesquisas dentro da área da trjetória, estendendo-se a representação do dado, o pré-processamento e as diversas

técnicas que possam ser aplicadas para detecção de padrões e anomalias.

- **Gomes *et al.* (2018)**: esse trabalho forneceu um complemento das operações expostas pelo trabalho de Zheng (2015) e a aplicação das mesmas voltadas para visualização dos dados de trajetórias. Além disso, forneceu uma base geral das atuais técnicas para gerar análises e visualizações desses dados.
- **Chen *et al.* (2015)** e **Gonçalves *et al.* (2013)**: esses trabalhos foram fundamentais para entender as técnicas de visualizações de dados espaço-temporais e os canais que podem ser explorados. Além disso, também forneceram os cenários onde cada técnica de visualização tem melhor aplicabilidade.
- **Pappalardo *et al.* (2019)**: esse foi o principal trabalho relacionado encontrado que alinha os objetivos e propostas com o PyMove.

4.1 Estruturação e modelagem da biblioteca

Inicialmente, o repositório do GitHub[1] do PyMove continha cento e onze funções distribuídas nos seis arquivos abaixo:

- ***gridutils.py***: funções relacionadas a criação de *grids* a partir de uma trajetória.
- ***maputils.py***: funções ligadas a visualizações dos dados espaços-temporais.
- ***mem.py***: funções ligadas a verificação de uso de memória e informações sobre processos.
- ***mem_usage.py***: com apenas uma função que verificava a quantidade de memória gasta por um objeto.
- ***osmutils.py***: funções ligadas a geração de grafos das ruas.
- ***trajutils.py***: funções ligadas a pré-processamento de trajetórias.
- ***utils.py*** : demais funções que serviam como utilidades e automatização de processos básicos no uso de trajetórias.

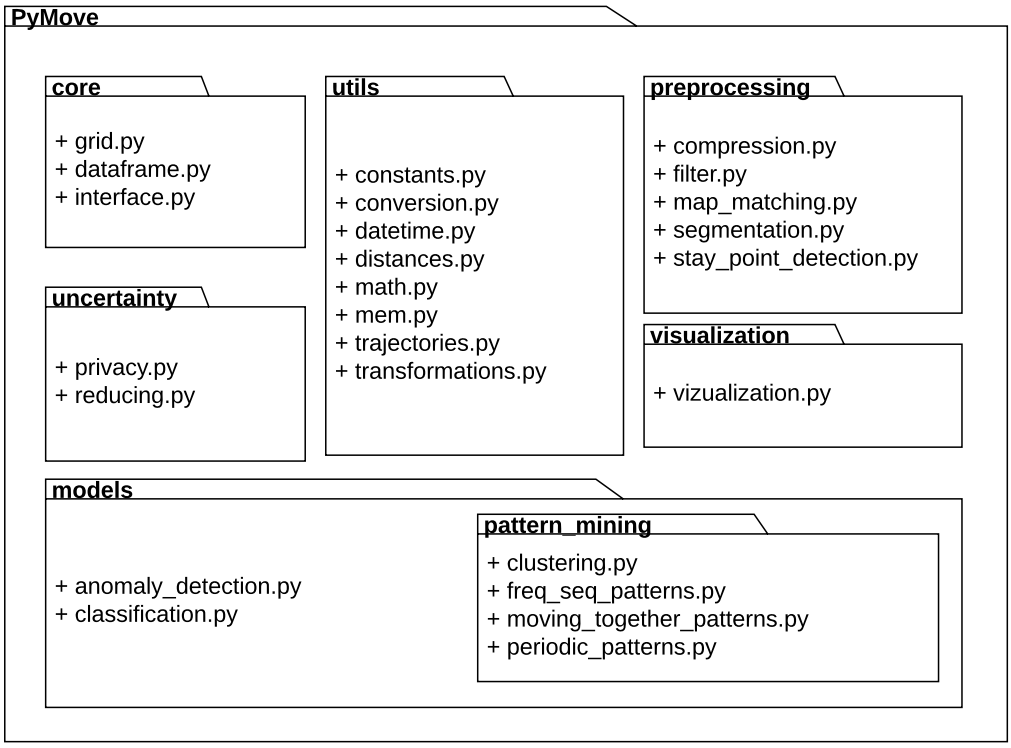
Após a etapa de revisão bibliográfica, foi realizada a modelagem da nova estrutura da biblioteca, apresentada na Figura 23. Essa modelagem é embasada na imagem original proposta por Zheng (2015), com as alterações apresentadas na Figura 1.

Na Figura 24, são mostrados os módulos existentes no pacote *models*. Devido nenhuma função previamente existente se adequar a nenhum desses módulos, eles estão temporariamente vazios. Esse pacote é constituído pelos módulos:

- ***anomaly_detection***: armazenará procedimentos que englobem técnicas de detecção de

[1] Repositório PyMove: https://github.com/InsightLab/PyMove

Figura 23 – Nova estrutura da biblioteca PyMove baseada na Figura 1.



Fonte: elaborada pelo autor.

anomalias, conforme descrito na Seção 2.1.4.2;

- ***classification***: armazenará procedimentos que englobem técnicas de classificação de trajetórias, conforme descrito na Seção 2.1.4.1;

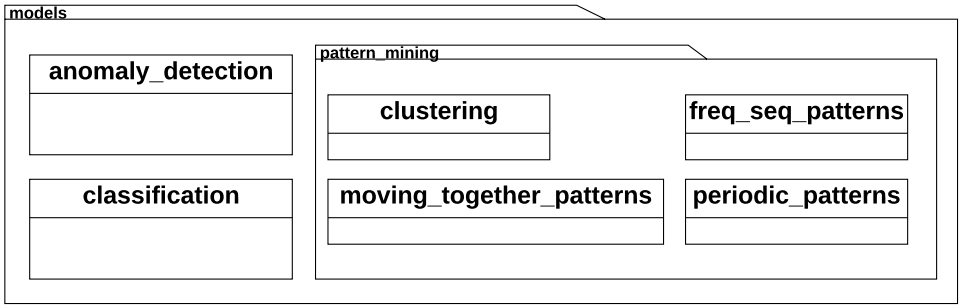
Além desses módulos, também há a presença do sub-pacote *patter_mining*, que descreve as técnicas apresentadas na Seção 2.1.4.3 nele há os módulos:

- ***clustering***: armazenará procedimentos que englobem técnicas de agrupamento de trajetórias;
- ***moving_together_patterns***: armazenará procedimentos que englobem técnicas de detecção de similaridade de movimento entre diferentes objetos;
- ***freq_seq_patterns***: armazenará procedimentos que englobem técnicas de detecção de padrões sequenciais entre trajetórias;
- ***periodic_patterns***: armazenará procedimentos que englobem técnicas de detecção de padrões temporais em trajetórias;

Na Figura 25, é mostrada a distribuição das funções que estavam antes no arquivo *trajutils.py* entre os módulos *segmetation*, *map_matching* e *filter*. Não há funções que existiam previamente não se adequavam a técnicas referentes aos módulos *compression* e *stay_point_detection*.

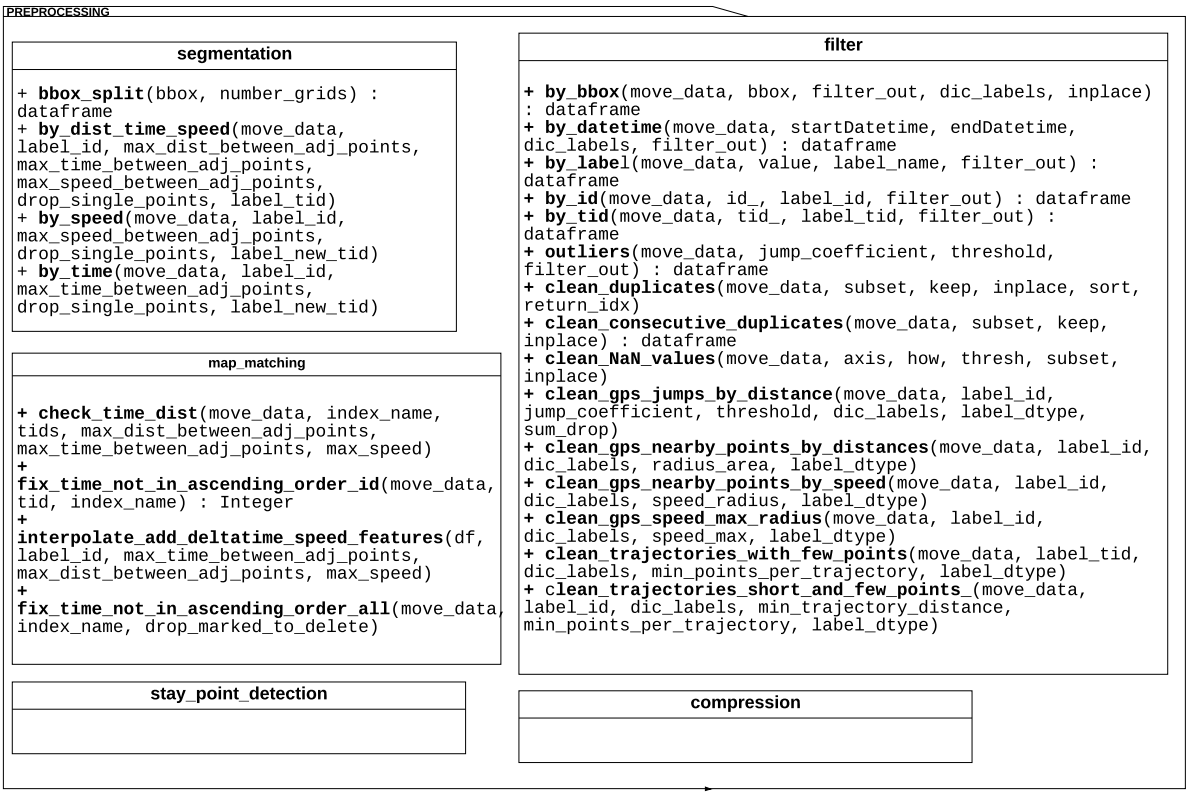
Na Figura 26, são mostrados os módulos existentes dentro do pacote *uncertainty*,

Figura 24 – Módulos presentes no pacote *models*.



Fonte: elaborada pela autora.

Figura 25 – Módulos presentes no pacote *preprocessing*.



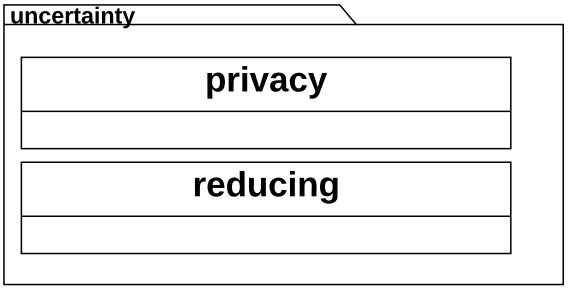
Fonte: elaborada pelo autor.

que contém os módulos *privacy*, *reducing*. Novamente, nenhuma função previamente existente se adequava a nenhum desses módulos.

Na Figura 27, é mostrado o único módulo existente dentro do pacote, o *visualization* e as funções presentes nesse módulo.

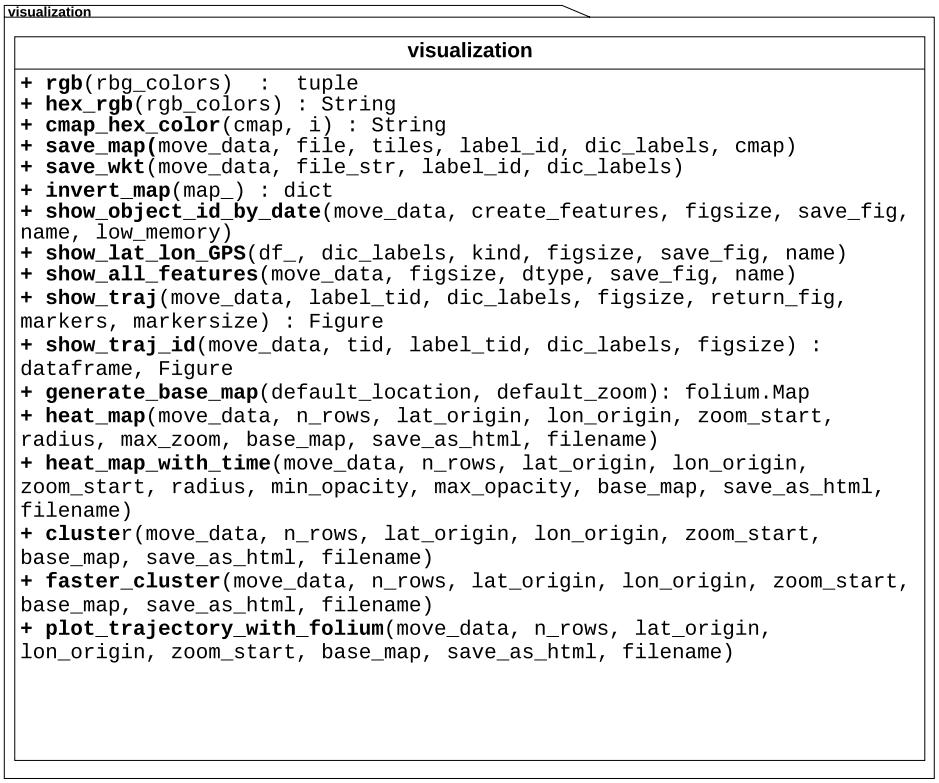
Na Figura 28, são mostrados os módulos existentes dentro do pacote *transformations*, *conversions*, *math*, *distances*, *datetime*, *constants* e *trajectories*, e a distribuição de funções entre esses.

Figura 26 – Módulos presentes no pacote *uncertainty*.



Fonte: elaborada pelo autor.

Figura 27 – Módulos presentes no pacote *visualization*.



Fonte: elaborada pelo autor.

4.1.1 Arquitetura das estruturas de dados da biblioteca

Após a modelagem da estrutura geral do *PyMove*, o propósito foi elaborar o escopo do núcleo (*core*) e a estrutura de dados da biblioteca, denominada *MoveDataFrame*. Foi idealizado que essa estrutura de dados fosse facilmente extensível e com sintaxe familiar aos usuários nativos do Pandas.

O primeiro item foi sanado através da modelagem da estrutura de dados utilizar o padrão *Factory Method*, apresentado na Seção 2.5. Esse padrão de projeto permite a existência de uma interface, o *MoveDataFrameAbstract*, que delimite o escopo que todas as outras extensões

Figura 28 – Módulos presentes no pacote *utils*.

Fonte: elaborada pelo autor.

com diferentes estruturas de dados devam prover, similar ao conceito do *Product*. A partir disso, há então os *ConcretProducts*, que são as classes que implementam o escopo delimitado pela interface anteriormente citada. Isso possibilita que haja vários *ConcretProducts* que utilizem bibliotecas, que proporcionem estruturas de dados e sua manipulação, abaixo da camada do *MoveDataFrame*. Com isso, neste trabalho, foram utilizados o Pandas e o Dask, apresentados nas Seções 2.3.1 e 2.3.2, para serem as estruturas de dados do *PandasMoveDataFrame* e *DaskMoveDataFrame*, respectivamente.

Diante desse cenário, a estrutura *MoveDataFrame* age como o *ConcretCreator*, que em seu construtor recebe além dos dados de entrada, uma *flag* referente a qual estrutura de dados o usuário quer utilizar. Assim, o construtor inicializa e retorna um objeto do tipo *PandasMoveDataFrame* ou *DaskMoveDataFrame*. O construtor age como o *FactoryMethod()*.

O ponto referente à sintaxe familiar do *MoveDataFrame* foi resolvido por meio de padronizar às funções da interface com assinaturas iguais ou semelhantes à API do Pandas.

4.2 Desenvolvimento

Esta seção apresenta os passos seguidos após a modelagem da biblioteca no âmbito da aplicação da modelagem e implementação das estruturas de dados e novas funções para o

módulo de visualizações do PyMove. Além disso, estão descritas as etapas de análise estática e depuração, documentação das funções e elaboração dos tutorias.

4.2.1 Implementação

Nesta seção é realizada a aplicação da modelagem definida na seção anterior. Além disso, são descritas as implementações das estruturas do *core* e do módulo de visualizações.

4.2.1.1 Implementação do Core

Após a definição da arquitetura das estrutura de dados da biblioteca, foi realizada a etapa de implementação das mesmas. Por limitações de tempo, o escopo deste trabalho limita-se ao desenvolvimento completo da classe *PandasMoveDataFrame* e o desenvolvimento parcial da classe *DaskMoveDataFrame* para validação da extensibilidade da ferramenta.

Para criação de um *MoveDataFrame* o dado deve possuir, obrigatoriamente, as colunas de latitude, longitude e data-hora, similar a estrutura TrajMoveDataFrame, apresentada na Seção 3.1. Além disso, o dado pode possuir colunas opcionais tais como: velocidade, id da trajetória e/ou do usuário, sendo que quando esses dois últimos não estão presentes nos dados assume-se que o dado apresenta somente uma trajetória associada a um objeto móvel.

A estrutura de dados instanciadas pelo *MoveDataFrame* permitem receber como dados de entrada as estruturas básicas do Python: as listas, dicionários, e arquivos csvs. Além disso, essas estruturas possuem implementação para gerar novas *features* a partir das colunas iniciais, como: velocidade, período do dia e dia da semana. Essas implementações podem ser visualizadas dentro do pacote *core* no repositório da biblioteca.

4.2.1.2 Implementação das visualizações

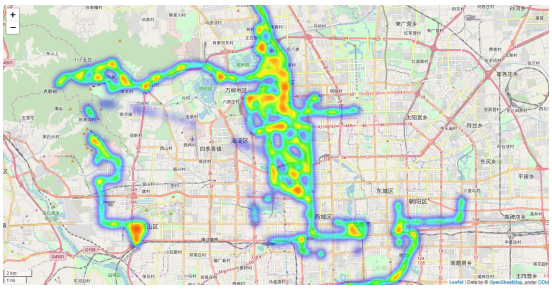
Após a implementação das estruturas de dados, foi realizada a implementação de novas funções que permitam gerar visualizações de dados. Elas são apresentadas na Figura 38. Neste módulo há funções que permitem e facilitam a geração de cores, permitem visualizar os dados de trajetória analisando características de tempo e, por fim, este módulo contém diversas visualizações que contemplam as técnicas destacadas na Seção 2.4. Estas funções são apresentadas e descritas no Apêndice A.

A Figura 38 mostra algumas das visualizações geradas pela biblioteca. A visualização

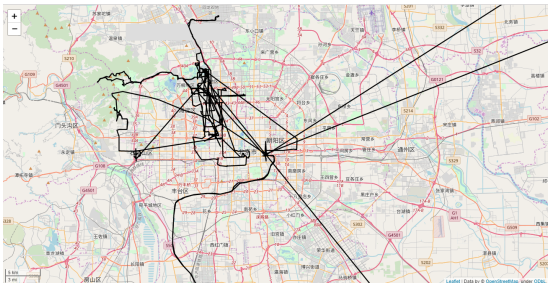
na Figura 29a é gerada pela função heat_map; a Figura 29b usa a função plot_trajectories_with_folium; já as Figuras 29c e 29d utilizam as funções cluster e faster_cluster, respectivamente.

Os códigos dessas visualizações podem ser encontrados no repositório da biblioteca no GitHub, no módulo *visualizations*.

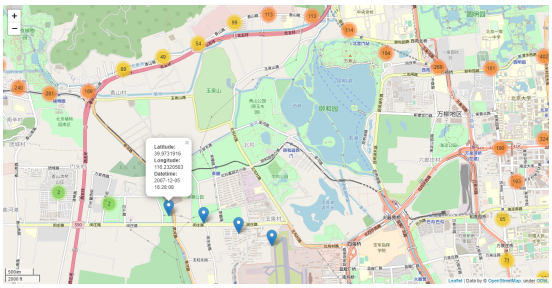
Figura 29 – Visualizações geradas pelas funções implementadas nessa etapa, que integram o módulo de visualizações.



(a) Visualização utilizando técnica Mapa de Calor, Seção 2.4.5



(b) Visualização de todos os pontos de todas trajetórias, utilizando técnica de Mapa estático, Seção 2.4.2



(c) Visualização utilizando técnica de *Cluster*, Seção 2.4.4 por meio do *plugin* MarkerCluster.

(d) Visualização utilizando técnica de *Cluster*, Seção 2.4.4 por meio do *plugin* FastMarkerCluster.

Fonte: elaborado pela autora

4.2.2 Análise estática e depuração das funções

Após a etapa de refatoração da biblioteca, implementação da arquitetura das estrutura de dados da biblioteca e novas visualizações, é realizada o procedimento de verificação das funções previamente existentes, por meio da análise estática do código. Essa etapa tem como objetivo corrigir possíveis *bugs*, tal como detectar e corrigir *code smells* nos códigos. Para isso foi utilizada a ferramenta PyLint, descrita na Seção 2.6.2, para maior precisão e correção dos problemas.

O padrão adotado para codificação e estilo do código foi a PEP8, descrita na Seção 2.6.1. As regras seguidas por este trabalho, são:

1. Utilizar quatro espaços para cada nível de indentação. Além disso, deve-se alinhar os

elementos quebrados verticalmente, alinhando-os entre parênteses, colchetes e chaves, ou usando um recuo suspenso;

2. Utilizar espaços invés de *tabs*. O Python 3 não permite misturar o uso de guias e espaços para recuo;
3. Todas as linhas devem possuir no máximo 79 caracteres;
4. Caso a linha possua mais que o limite máximo de caracteres e se a mesma possuir uma operação matemática, a mesma deve ser quebrada antes dos operadores matemáticos;
5. Deve-se utilizar duas linhas em branco para definição de funções e classe de nível superior, e uma linha para separar métodos de uma classe;
6. Importações na parte superior do arquivo e em linhas separadas;
7. Não utilizar espaços em branco antes da vírgula ponto-e-vírgula, dois pontos, entre colchetes, parênteses ou chaves; entre vírgula à direita e um parêntese em seguida
8. Convenções de nomenclatura, onde usa-se as palavras minúscula separadas por *under score* para nome de funções e variáveis.

4.2.3 Documentação das funções

Esta etapa consiste em prover documentação as funções da biblioteca, tal como elaborar *notebooks* do Jupyter[2] a fim de exemplificar o seu uso.

Todas as classes, funções e métodos da biblioteca foram documentadas, logo após a assinatura. Essa documentação segue o padrão de bibliotecas reconhecidas dentro da comunidade do Python: Scikit-Learn e Pandas. Nesse padrão são identificados:

- Objetivo da função;
- Parâmetro da função, o seu tipo e o que representa;
- Retorno da função, o tipo e o que representa;
- Exemplo de aplicação da função;
- Referências para elaboração da função;
- Notas.

4.2.4 Elaboração de Tutoriais

Nesta etapa foram produzidos *jupyter notebooks*, com a finalidade de servirem como guias e tutoriais para novos usuários da ferramenta. Neles são descritas as finalidade do uso e

[2] https://jupyter.org/

exploradas as funções presentes em cada módulo.

No total são quatro tutoriais, sendo dois elaborados neste trabalho: *Exploring MoveDataFrame* e *Exploring Visualizations*. Estes tutoriais podem ser encontrados na pasta *examples*[3] do repositório da biblioteca no GitHub.

Para elaboração dos tutoriais, o conjunto de dados utilizado neste trabalho foi o *Geolife GPS trajectory dataset*, Zheng *et al.* (2011). Ele é resultado da coleta realizada no projeto Geolife, da Microsoft Research Asia, por 178 usuários por mais de quatro anos. Esses dados registram uma grande variedade de movimentos de dos usuários, incluindo não só caminhos rotineiros, como também atividades esportivas e de lazer, amplamente distribuídos em mais de 30 cidades da China e em algumas cidades localizadas nos EUA e na Europa.

Esse conjunto de dados possui 17.621 trajetórias, registradas por GPS, com uma distância total de 1.251.654 quilômetros e duração de 48.203 horas. Os dados estão dispostos em 182 pastas, numeradas de 000 à 181, onde cada uma simbolizava um usuário. Dentro de cada pasta, há uma pasta *Trajectory*, que contém as trajetórias GPS dos usuários em arquivos com extensão *plt*. Em algumas pastas de usuário, além da pasta *Trejctory*, também um arquivo *labels.txt* que informa o tipo de transporte utilizado pelo usuário em determinados períodos de tempo. Os arquivos que contém os dados de localiação possuem os valores separados por vírgula, onde representam, respectivamente, a latitude, longitude, campo sem valor, altitude, número de dias que passaram após a data 30/12/1999, data e tempo.

Por limitações de processamento e em busca de diversidade, nesse trabalho são utilizados os dados dos usuários 000, 010, 011 e 100, totalizando em 1.206.506 pontos de trajetórias.

4.3 Validação da biblioteca: Estudo de Caso

Para validar o uso da ferramenta, nesta seção é desenvolvido o Estudo de Caso. Este Estudo de Caso serve para validar a estrutura de dados elaborada para o PyMove, tal como apresentar como as as técnicas de visualizações presentes na biblioteca podem auxiliar no processo de análise dos dados. O conjunto de dados utilizado para a realização do Estudo de Caso é o *Geolife GPS trajectory dataset*, descrito na Seção 4.2.4.

[3] https://github.com/InsightLab/PyMove/tree/developer

4.3.1 Configuração do ambiente

Para utilizar o PyMove é necessário possuir na máquina o Python, com versão maior igual a 2.7, e o gerenciador de dependências pip. Com isso, deve-se seguir os seguintes passos:

1. Baixar o repositório, através do comando *git clone https://github.com/InsightLab/PyMove*;
2. Criar uma *branch developer*, através do comando *git branch developer*;
3. Mover-se para a *branch* criada, através do comando *git checkout developer*;
4. Obter todas as alterações dessa *branch*, através do comando *git pull origin developer*;
5. Mover-se para a pasta PyMove, através do comando *cd PyMove*.
6. Obter as dependências da biblioteca, através do comando *python setup.py install*.

Após a execução desses passos, para uar a biblioteca basta utilizar a seguinte linha de código: *import pymove*.

4.3.2 Validando MoveDataFrame

A importação dos dados pode ser realizada:

- Pela leitura direta de um arquivo, conforme ilustrado pela Figura 30;
- Por uma lista do Python, conforme mostrado na Figura 31;
- Por um dicionário do Python, conforme mostrado na Figura 33;
- Por um DataFrame do Pandas, Seção 2.3.1, conforme ilustrado pela Figura 32.

Figura 30 – Importação dos dados utilizando leitura de arquivo.

```
In [2]: from pymove import MoveDataFrame

        move_df = pymove.read_csv('examples/geolife.csv')
        move_df.head()
```

Out[2]:

	Unnamed: 0	datetime	lat	lon	alt	id	user
0	0	2008-10-23 02:53:04	39.984702	116.318417	492.0	0	0
1	1	2008-10-23 02:53:10	39.984683	116.318450	492.0	0	0
2	2	2008-10-23 02:53:15	39.984686	116.318417	492.0	0	0
3	3	2008-10-23 02:53:20	39.984688	116.318385	492.0	0	0
4	4	2008-10-23 02:53:25	39.984655	116.318263	492.0	0	0

Fonte: elaborado pela autora

Por *default*, o dado criado é uma instância de um *PandasMoveDataFrame*. Isso pode ser alterado por meio da passagem de uma *flag* na instanciação da estrutura. Também é permitido fazer conversões de uma estrutura para outra, como de um *PandasMoveDataFrame* para um

Figura 31 – Importação dos dados via estrutura lista do Python.

```
In [3]: from pymove import MoveDataFrame

        list_data = [[39.984094, 116.319236, '2008-10-23 05:53:05', 1],
                     [39.984198, 116.319322, '2008-10-23 05:53:06', 1],
                     [39.984224, 116.319402, '2008-10-23 05:53:11', 1],
                     [39.984224, 116.319402, '2008-10-23 05:53:11', 1],
                     [39.984224, 116.319402, '2008-10-23 05:53:11', 1],
                     [39.984224, 116.319402, '2008-10-23 05:53:11', 1]]
        move_df = MoveDataFrame(data=list_data)
        move_df.head()
```

Out[3]:

	lat	lon	datetime	id
0	39.984094	116.319236	2008-10-23 05:53:05	1
1	39.984198	116.319322	2008-10-23 05:53:06	1
2	39.984224	116.319402	2008-10-23 05:53:11	1
3	39.984224	116.319402	2008-10-23 05:53:11	1
4	39.984224	116.319402	2008-10-23 05:53:11	1

Fonte: elaborado pela autora

Figura 32 – Importação dos dados via DataFrame do Pandas.

```
In [1]: import pandas as pd
        from pymove import MoveDataFrame

        df = pd.read_csv('examples/geolife.csv')
        move_df = MoveDataFrame(data=df)

        move_df.head()
```

Out[1]:

	Unnamed: 0	datetime	lat	lon	alt	id	user
0	0	2008-10-23 02:53:04	39.984702	116.318417	492.0	0	0
1	1	2008-10-23 02:53:10	39.984683	116.318450	492.0	0	0
2	2	2008-10-23 02:53:15	39.984686	116.318417	492.0	0	0
3	3	2008-10-23 02:53:20	39.984688	116.318385	492.0	0	0
4	4	2008-10-23 02:53:25	39.984655	116.318263	492.0	0	0

Fonte: elaborado pela autora

Figura 33 – Importação dos dados via estrutura dicionário do Python.

```
In [4]: from pymove import MoveDataFrame

        dict_data = {
            'lat': [39.984198, 39.984224, 39.984094],
            'lon': [116.319402, 116.319322, 116.319402],
            'datetime': ['2008-10-23 05:53:11', '2008-10-23 05:53:06', '2008-10-23 05:53:06']
        }

        move_df = MoveDataFrame(data=dict_data)
        move_df.head()
```

Out[4]:

	lat	lon	datetime
0	39.984198	116.319402	2008-10-23 05:53:11
1	39.984224	116.319322	2008-10-23 05:53:06
2	39.984094	116.319402	2008-10-23 05:53:06

Fonte: elaborado pela autora

DaskMoveDataFrame e vice-versa. Esses detalhes podem ser vistos nos guias e tutoriais criados em uma das etapas anteriores.

4.3.3 Validando as Visualizações

As visualizações implementadas no PyMove descritas acima permitem extrair informações e realizar análises visuais. Para auxiliar o usuário final na escolha de cores e ir além das cores *default* reconhecidas por *strings*, para visualizações, o PyMove apresenta as funções exibidas na Figura 34, já comentadas anteriormente.

Figura 34 – Resultado das funções que geram cores do PyMove.

```
In [9]:  visualization.generate_color()
Out[9]:  '#FF7F50'

In [10]: rgb = visualization.rgb([0.6, 0.2, 0.2])
         rgb
Out[10]: (51, 51, 153)

In [11]: visualization.hex_rgb([0.6, 0.2, 0.2])
Out[11]: '#333399'
```

Fonte: elaborado pela autora

Para realizar análises de características de uma trajetória pelo tempo utilizando o PyMove, pode-se utilizar a função *show_object_id_by_date*. A Figura 35 mostra a visualização gerada pela execução desta função. Nela são criadas novas *features* baseadas nas originais.

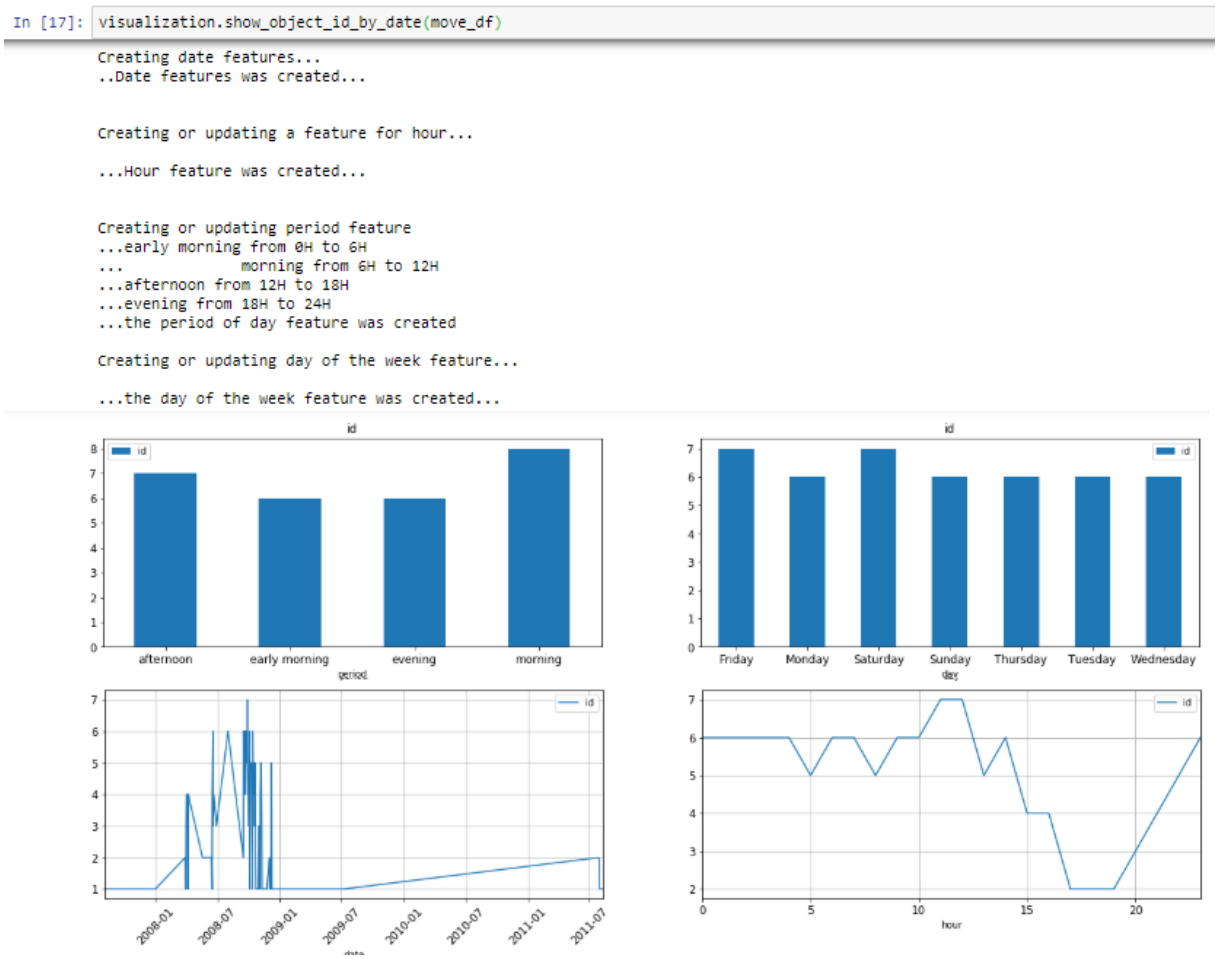
Para visualizar todos os dados de trajetórias e ter uma visão geral do que os dados apresentam pode ser utilizado a função *plot_trajectories_with_folium*, conforme ilustrado na Figura 36a. Já para visualizar uma trajetória pelo seu identificador pode ser utilizada a função *plot_trajectory_by_id_with_folium*, conforme ilustrado na Figura 36b, onde são utilizados os primeiros 1110 pontos dessa trajetória.

O PyMove também oferece funções que permitem visualizações de trajetórias por dia da semana, como demonstrado na Figura 36c, e por data, Figura 36d.

Para visualizar quais as rotas quentes desses dados pode ser utilizado a função *heatmap*, que utiliza a técnica de Mapa de Calor, 2.4.5. A Figura 37 demonstra o resultado obtido analisando cinquenta e quatro mil pontos. Nela as regiões avermelhadas demonstra o forte fluxo, enquanto regiões mais azuladas possuem um fluxo não tão intenso.

Visto o tamanho do conjunto de dados utilizado, reproduzir uma visualização que contenha todos os pontos marcados no mapa possivelmente não será eficiente, em razão da poluição visual gerada, ilustrado na Figura 38a. Com a utilização da função *cluster* pode ser

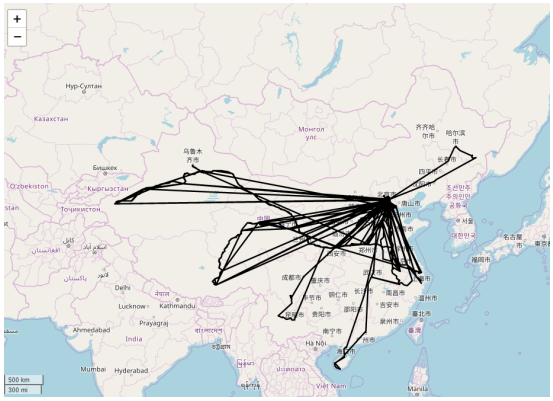
Figura 35 – Visualizações geradas pela função *show_object_id_by_date*, explorando técnica de Mapa estático, Seção 2.4.2



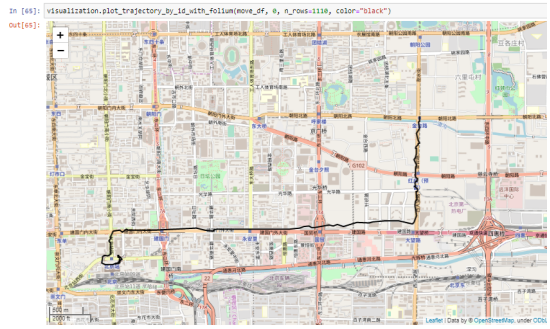
Fonte: elaborado pela autora

resolvido o problema de poluição, ilustrada na Figura 38b. Além disso os grupos formatos interativamente fornecem ao usuário o entendimento dos locais com maior acúmulo de pontos. Esse acúmulo de pontos em um local pode ser um forte indício que este local é um ponto de parada. Após o clique no grupo que contém cento e oito pontos da Figura 38b, a Figura 38c ilustra a espiral formada pelos pontos. Além disso, nesta figura também é possível verificar o tempo inicial do ponto neste local no dia 23-10-2008. Já a Figura 38d demonstra o último horário do dia 23-10-2008 que este objeto ficou neste local. A diferença de horários nos revela que esse ponto é um forte candidato a ponto de parada.

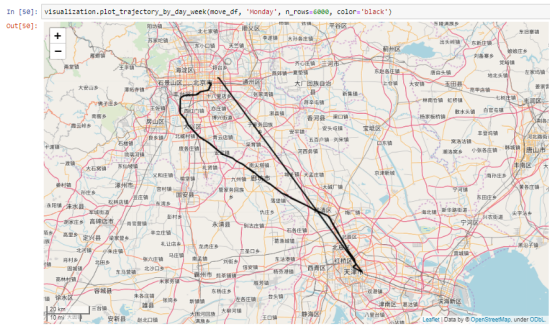
Figura 36 – Visualizações geradas pelas funções que exploram a técnica de Mapa estáticos, Seção 2.4.2, que integram o módulo de visualizações.



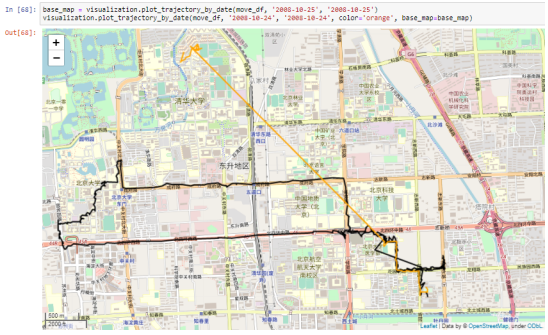
(a) Visualização de todas as trajetórias.



(b) Visualização da trajetória com identificador 0.



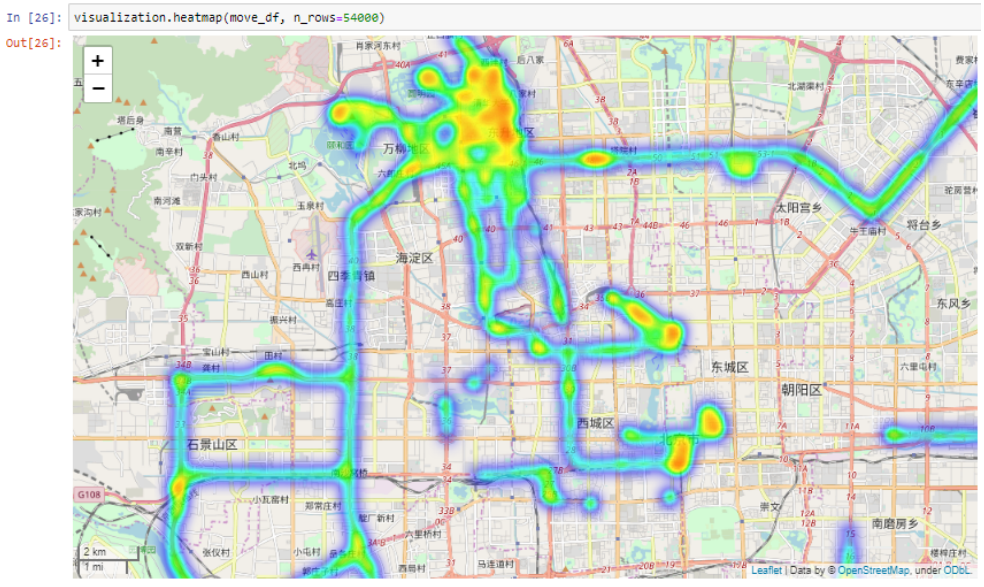
(c) Visualização das trajetórias em uma Segunda-Feira.



(d) Visualização das trajetórias do dia 24 e 25 de outubro de 2008.

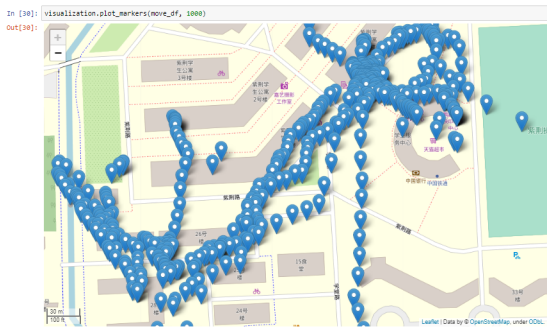
Fonte: elaborado pela autora

Figura 37 – Resultado da aplicação da função *heatmap* sobre os dados *Geolife GPS trajectory*, técnica Mapa de Calor, Seção 2.4.5

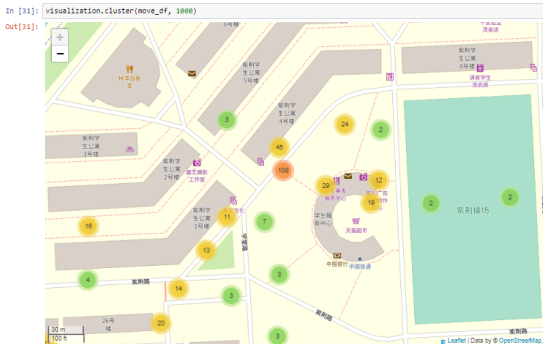


Fonte: elaborado pela autora

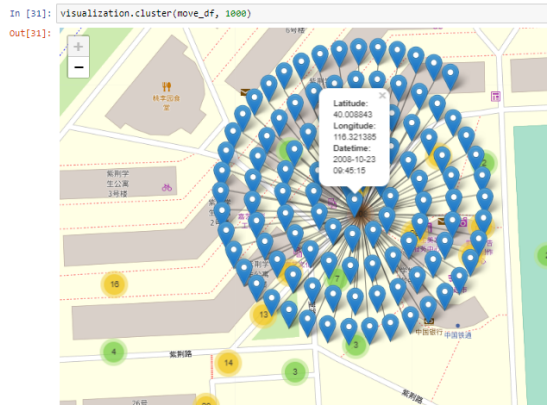
Figura 38 – Visualizações geradas pelas funções implementadas nessa etapa, que integram o módulo de visualizações.



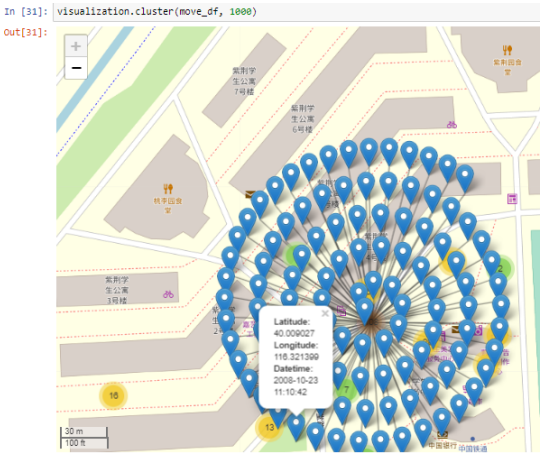
(a) Visualização de marcadores que representam os pontos do conjunto de dados.



(b) Visualização da mesma quantidade de pontos utilizando a função *cluster*.



(c) Horário inicial de um objeto móvel em um ponto no dia 23-10-2008.



(d) Horário final de um objeto móvel em um ponto no dia 23-10-2008.

Fonte: elaborado pela autora

5 CONSIDERAÇÕES FINAIS

Neste trabalho foi apresentado o processo de desenvolvimento, indo desde a estruturação e proposta de uma modelagem para a biblioteca *PyMove*, baseada na organização proposta em Zheng (2015), a verificação estática dos códigos, documentação das funções e gerando *notebooks Jupyter* para exemplificar o uso das mesmas. Além disso, foi também apresentada as novas funções adicionadas ao módulo de visualização da biblioteca.

O trabalho aqui proposto, possui o objetivo de propor a modelagem da biblioteca PyMove e incorporar técnicas de visualizações no módulo dessa biblioteca, a fim de amenizar o deficit de ferramentas para trabalhar com esse tipo de dado, conforme apresentado no início desse documento.

As vantagens de se utilizar o *PyMove* consistem principalmente em:

- Oferecer uma maior produtividade no processo de pesquisa e trabalho com dados espaço-temporais, visto que, sem ela, o usuário/indivíduo necessitaria de adaptar os algoritmos dispostos na literatura e implementa-los por si mesmo;
- Variedade de técnicas de visualizações para análises visuais;
- Facilidade de extensão, seja na adição de novos módulos ou o uso de outras bibliotecas que forneçam suporte para manipulação de dados, como o Modin (PETERSOHN; JOSEPH, 2018), para comportar necessidades dos usuários;
- Possibilidade de tornar a biblioteca escalável frente a uma grande quantidade de dados, através do implementação do *MoveDataFrame* utilizando por baixo a estrutura de dados do Dask, a biblioteca pode comportar computação paralela e as demais vantagens advindas do Dask e esse paradigma.

Estas vantagens exploram os pontos fortes e fracos presentes nos trabalhos relacionados analisados, apresentados no Capítulo 3, combinando-os, conforme apresentado na Seção 3.4. Em contrapartida, a biblioteca, apesar de ter considerado os fatores na modelagem, ainda não possui implementação de técnicas no âmbito de clusterização, mineração de padrões e privacidade dos dados do usuário.

Em detrimento a esses pontos, visa-se que trabalhos futuros possam atacar os seguintes pontos:

- Explorar e implementar novas formas de análises visuais para incrementar o módulo de visualizações. Como, por exemplo, em Pappalardo *et al.* (2019), que é feita a visualização dos fluxos, dos pontos de origem e destino, visualização dos pontos de parada e o uso da

técnica *Choropleth*, presente na Seção 2.4.6. Além disso, conforme os outros módulos evoluam, poder gerar visualizações ligadas a eles;

- Estender a implementação da estrutura *DaskMoveDataFrame*, que utiliza o Dask como a camada inferior de estrutura de dados. Neste trabalho essa estrutura foi apenas inicializada e implementada algumas funções para validar a extensibilidade da ferramenta. Com a finalização dessa estrutura, visa-se permitir a flexibilidade para o usuário final, que, por sua vez, poderá migrar de uma estrutura a outra e trabalhar com aquela que melhor atende às suas necessidades;
- Expandir a biblioteca através da implementação de técnicas para os demais módulos, visando cobrir uma maior variedade de aspectos relacionados à análise de dados de mobilidade. Essa implementação se diz respeito principalmente aos módulos de *uncertainty*, que diz respeito a privacidade dos dados, tanto para redução quanto aumento das incertezas dos dados; e para o módulo *models*, que diz respeito as técnicas que possam ser aplicados nos dados após pré-processamento, englobando a clusterização, detecção de anomalias e mineração de padrões.

REFERÊNCIAS

ANDRIENKO, N.; ANDRIENKO, G. Visual analytics of movement: An overview of methods, tools and procedures. **Information Visualization**, v. 12, n. 1, p. 3–24, 2013. Disponível em: <https://doi.org/10.1177/1473871612457601>.

BATISTA, L. F. A.; IMAI, N. N.; ROTTA, L. H. d. S.; WATANABE, F. S. Y.; VELINI, E. D. Mineração de dados espaço-temporais para reconhecimento de padrões relacionados à vegetação aquática submersa. **Anais XV Simpósio Brasileiro de Sensoriamento Remoto - SBSR, Curitiba, PR, Brasil,**, n. 2005, p. 4752–4759, 2011. ISSN 00223077.

CHEN, W.; GUO, F.; WANG, F.-Y. A survey of traffic data visualization. **IEEE Transactions on Intelligent Transportation Systems**, v. 16, p. 2970–2984, 12 2015.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design Patterns: Elements of Reusable Object-oriented Software**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995. ISBN 0-201-63361-2.

Gomes, G. A. M.; Santos, E.; Vidal, C. A. **VISUALIZAÇÃO INTERATIVA DE DINÂMICAS DE TRÁFEGO ATRAVÉS DE DADOS DE TRAJETÓRIAS**. Tese (Doutorado) — PD thesis, Universidade Federal do Ceará, 2018.

GONçALVES, T.; AFONSO, A.; MARTINS, B. Visualization techniques of trajectory data: Challenges and limitations. **CEUR Workshop Proceedings**, v. 1136, 01 2013.

HA, Y. W.; MOON, J.-Y.; JUNG, H.-J.; CHUNG, B. C.; CHOI, M. H. Evaluation of plasma enzyme activities using gas chromatography–mass spectrometry based steroid signatures. **Journal of Chromatography B**, v. 877, n. 32, p. 4125 – 4132, 2009. ISSN 1570-0232. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1570023209007727>.

HOREMUŽ, M.; ANDERSSON, J. V. Polynomial interpolation of gps satellite coordinates. **GPS Solutions**, v. 10, n. 1, p. 67–72, Feb 2006. ISSN 1521-1886. Disponível em: <https://doi.org/10.1007/s10291-005-0018-0>.

HUNTER, J. D. Matplotlib: A 2d graphics environment. **Computing in Science & Engineering**, IEEE COMPUTER SOC, v. 9, n. 3, p. 90–95, 2007.

HWANG, S.; VANDEMARK, C.; DHATT, N.; YALLA, S. V.; CREWS, R. T. Segmenting human trajectory data by movement states while addressing signal loss and signal noise. **International Journal of Geographical Information Science**, Taylor '&' Francis, v. 32, n. 7, p. 1391–1412, 2018. Disponível em: <https://doi.org/10.1080/13658816.2018.1423685>.

KARIMI, H.; KRISHNAMURTHY, P.; PELECHRINIS, K. Location-Based Social Networks. **Advanced Location-Based Technologies and Services**, p. 127–144, 2013.

MAGALHÃES, R. **SPEED PREDICTION APPLIED TO DYNAMIC TRAFFIC SENSORS AND ROAD NETWORKS**. Tese (Doutorado) — PD thesis, Universidade Federal do Ceará, 2018.

MATEJKA, J.; GROSSMAN, T.; FITZMAURICE, G. Patina: Dynamic heatmaps for visualizing application usage. **Conference on Human Factors in Computing Systems - Proceedings**, p. 3227–3236, 2013.

MCKINNEY, W. pandas: a foundational python library for data analysis and statistics. **Python High Performance Science Computer**, 01 2011.

METSALU, T.; VILO, J. ClustVis: A web tool for visualizing clustering of multivariate data using Principal Component Analysis and heatmap. **Nucleic Acids Research**, v. 43, n. W1, p. W566–W570, 2015. ISSN 13624962.

MORSHED, A.; TSAI, P.-W.; JAYARAMAN, P. P.; SELLIS, T.; GEORGAKOPOULOS, D.; BURKE, S.; JOACHIM, S.; QUAH, M.-S.; TSVETKOV, S.; LIEW, J.; JENKINS, C. Viscrime: A crime visualisation system for crime trajectory from multi-dimensional sources. In: **Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining**. New York, NY, USA: ACM, 2019. (WSDM '19), p. 802–805. ISBN 978-1-4503-5940-5. Disponível em: <http://doi.acm.org/10.1145/3289600.3290617>.

PAPPALARDO, L.; SIMINI, F.; BARLACCHI, G.; PELLUNGRINI, R. **scikit-mobility: a Python library for the analysis, generation and risk assessment of mobility data**. 2019.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V. *et al.* Scikit-learn: Machine learning in python. **Journal of machine learning research**, v. 12, n. Oct, p. 2825–2830, 2011.

PETERSOHN, D.; JOSEPH, A. D. Scaling Interactive Data Science Transparently with Modin. 2018.

REY, S.; ANSELIN, L. Pysal: A python library of spatial analytical methods. **Review of Regional Studies**, Oklahoma State University, v. 37, n. 1, p. 5–27, 2007. ISSN 1553-0892.

ROCKLIN, M. Dask: Parallel Computation with Blocked algorithms and Task Scheduling. **Proceedings of the 14th Python in Science Conference**, n. SCIPY, p. 126–132, 2015.

RUAN, S.; LI, R.; BAO, J.; HE, T.; ZHENG, Y. Cloudtp: A cloud-based flexible trajectory preprocessing Framework. **Proceedings - IEEE 34th International Conference on Data Engineering, ICDE 2018**, p. 1601–1604, 2018.

SACHARIDIS, D.; PATROUMPAS, K.; TERROVITIS, M.; KANTERE, V.; POTAMIAS, M.; MOURATIDIS, K.; SELLIS, T. On-line discovery of hot motion paths. In: **Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology**. New York, NY, USA: ACM, 2008. (EDBT '08), p. 392–403. ISBN 978-1-59593-926-5. Disponível em: <http://doi.acm.org/10.1145/1353343.1353392>.

van der Walt, S.; Colbert, S. C.; Varoquaux, G. The numpy array: A structure for efficient numerical computation. **Computing in Science Engineering**, v. 13, n. 2, p. 22–30, March 2011. ISSN 1558-366X.

WEI, L. Y.; ZHENG, Y.; PENG, W. C. Constructing popular routes from uncertain trajectories. **Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, p. 195–203, 2012.

ZAHARIA, M.; CHOWDHURY, M.; FRANKLIN, M.; SHENKER, S.; STOICA, I. Spark: Cluster computing with working sets. **Proceedings of the 2nd USENIX conference on Hot topics in cloud computing**, v. 10, p. 10–10, 07 2010.

ZHENG, Y. Trajectory data mining: An overview. **ACM Transaction on Intelligent Systems and Technology**, September 2015. Disponível em: <https://www.microsoft.com/en-us/research/publication/trajectory-data-mining-an-overview/>.

ZHENG, Y.; FU, H.; XIE, X.; MA, W.-Y.; LI, Q. GeoLife User Guide 1.2. **Microsoft Reasearch Asia**, v. 2, n. April 2007, p. 31–34, 2011. Disponível em: <https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide/>.

ZHENG, Y.; LIU, T.; WANG, Y.; ZHU, Y.; LIU, Y.; CHANG, E. Diagnosing New York city's noises with ubiquitous data. **UbiComp 2014 - Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing**, p. 715–725, 2014.

APÊNDICE A – MÓDULO DE VISUALIZAÇÃO DO PYMOVE

Este apêndice apresenta as funções presentes no módulo de visualização da biblioteca PyMove. O Quadro 2 demonstra as funções voltadas para gerar cores, a fim de auxiliar o usuário final no processo de geração de suas análises. O Quadro 4 apresentam as funções voltadas para implantação das técnicas de mapa de calor, mapas estáticos, *cluster* e gráfico de linhas, descritas em 2.4.

Além das funções presentes no módulo de visualização, também há as funções presentes na própria estrutura de dados, apresentadas no Quadro 3.

Quadro 2 – Funções para gerar cores.

Função	**Descrição**
rgb()	Retorna uma tupla com 3 números inteiros que variam de 0 a 255, onde estes números representam a intensidade das cores vermelho, verde e azul, respectivamente.
generate_color()	Gera e retorna cores aleatórias, a fim de fornecer uma gama maior de opções de cores em suas visualizações.
hex_rgb()	A partir de uma tupla contendo a intensidade das cores, passada como parâmetro, onde cada posição varia de 0 a 1, retorna uma cor hexadecimal no formato de string correspondente.
cmap_hex_color	Converte um mapa de cores em cores hexadecimais.

Fonte: elaborado pelo autor.

Quadro 3 – Funções para visualizações na estrutura MoveDataFrame.

Função	**Descrição**
plot_all_features()	Gera uma visualização, utilizando o Matplotlib, para cada coluna do conjunto de dados passado como parâmetro, visando visualizar de modo geral o comportamento das características conhecidas dos dados.
plot_trajs()	Gera uma visualização, utilizando o Matplotlib, para observar todas as trajetórias presente no conjunto de dados passado como parâmetro.
plot_traj_id()	Gera uma visualização, utilizando o Matplotlib, que mostra uma trajetória com seu ponto inicial e final, com um identificador informado pelo usuário final.

Fonte: elaborado pelo autor.

Quadro 4 – Funções para gerar visualizações.

Função	**Descrição**
show_object_id_by_date()	Gera quatro visualizações, utilizando o Matplotlib, com base no recurso de data e hora: - Trajetórias do gráfico de barras por períodos do dia; - Trajetórias do gráfico de barras dia da semana; - Trajetória do gráfico de linhas por data; - Gráfico de linhas da trajetória por horas do dia.
show_lat_lon_GPS()	Gera uma visualização, utilizando o Matplotlib, com as coordenadas dos pontos de latitude e longitude do conjunto de dados a fim de identificar o contexto geral dos dados.
create_base_map()	Gera um mapa do Folium centrado de acordo com as coordenadas passadas como parâmetro.
heatmap()	Gera uma visualização utilizando a técnica Mapa de Calor por meio do plugin HeatMap do Folium.
cluster()	Gera uma visualização utilizando a técnica Cluster por meio do plugin MarkerCluster do Folium, onde os pontos são simbolizados por meio de um marcador.
faster_cluster()	Também gera uma visualização utilizando a técnica Cluster, visando maior desempenho. Faz uso do plugin FastMarkerCluster do Folium, onde os pontos são simbolizados por meio de um círculo.
plot_markers()	Gera uma visualização com marcadores representando os dados passados como parâmetro.
plot_trajectory_with_folium()	Gera uma visualização utilizando o Folium, por meio linhas para representar todas as trajetórias presentes no conjunto de dados passado como parâmetro.
plot_trajectory_by_id_with_folium()	Gera uma visualização utilizando o Folium, por meio linhas para representar a trajetória com um identificador especificado pelo usuário final presente no conjunto de dados passado como parâmetro.
plot_trajectory_by_period()	Gera uma visualização utilizando o Folium, por meio linhas para representar as trajetórias delimitadas por um período do dia, especificado pelo usuário final, presente no conjunto de dados passado como parâmetro.
plot_trajectory_by_day_week()	Gera uma visualização utilizando o Folium, por meio linhas para representar as trajetórias delimitadas pelo dia da semana, especificado pelo usuário final, presente no conjunto de dados passado como parâmetro.
plot_trajectory_by_date()	Gera uma visualização utilizando o Folium, por meio linhas para representar as trajetórias delimitadas por uma data inicial e final, especificado pelo usuário final, presente no conjunto de dados passado como parâmetro.
plot_trajectory_by_hour()	Gera uma visualização utilizando o Folium, por meio linhas para representar as trajetórias delimitadas por um horário inicial e final, especificado pelo usuário final, presente no conjunto de dados passado como parâmetro.
plot_stops()	Gera uma visualização contendo os pontos de paradas do conjunto de dados passado como parâmetro. Estes são representados como círculos. A detecção desses pontos de paradas é feita por meio de funções do módulo de stay_point_detection do pacote de preprocessing.

Fonte: elaborado pelo autor.