Graduação em Ciência da Computação

Pedro Lins de Araujo Lima

UM ESTUDO SOBRE A COMPUTAÇÃO APROXIMADA DA PROFUNDIDADE DE BANDA E SUA APLICAÇÃO PARA A VISUALIZAÇÃO DE GRANDES COLEÇÕES DE SÉRIES TEMPORAIS

Trabalho de Graduação

Universidade Federal de Pernambuco
secgrad@cin.ufpe.br
www.cin.ufpe.br/~secgrad

RECIFE
2019

Universidade Federal de Pernambuco
Centro de Informática
Graduação em Ciência da Computação

Pedro Lins de Araujo Lima

UM ESTUDO SOBRE A COMPUTAÇÃO APROXIMADA DA PROFUNDIDADE DE BANDA E SUA APLICAÇÃO PARA A VISUALIZAÇÃO DE GRANDES COLEÇÕES DE SÉRIES TEMPORAIS

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: *Nivan Roberto Ferreira Júnior*

RECIFE
2019

À memória dos meus avós: Creusa, Jorge, Lourdes e Luiz.

Agradecimentos

Aos meus pais, Nadja e Jorge, e à minha irmã, Fernanda, pelo apoio incondicional e por estarem comigo desde o começo, incentivando a minha curiosidade e me encorajando a traçar meu próprio caminho.

À minha namorada, Camila, por me incentivar e ajudar sempre, por me ouvir e me entender indepedente de qualquer coisa e também por me encher de alegria a cada momento.

Aos meus primos e padrinhos, Lauro e Isis, por serem exemplos para mim e por me ajudarem sempre que eu precisei, independente do fuso horário.

À minha familia: vovó Lourdes, vovô Luiz, Laura, Mateus, Bernardete, Sóstenes, Niedja, Mariana, Arthur, Sofia, Vicente; vovó Creusa, vovô Jorge, tia Iracema, tia Baia, tio Mário, tia Salette, tia Socorrinho e todos os meus primos.

Aos meus amigos, por transformarem diversos momentos de tensão e nervosismo durante a graduação em momentos de alegria e risadas.

A todos do Laboratório OKI-Brasil pela paciência e por todo o aprendizado que me proporcionaram.

Ao meu professor e orientador Nivan, por trabalhar comigo neste projeto, bem como por estar disponível para tirar dúvidas e dar conselhos sempre que necessário.

Ao Centro de Informática e todos os seus funcionários e colaboradores por propiciarem um ambiente proveitoso para a construção de todo o meu aprendizado e para o desenvolvimento deste trabalho.

All we have to decide is what to do with the time that is given us.

—J. R. R. TOLKIEN

Resumo

Séries temporais são uma das formas mais comuns de registro de dados, sendo amplamente utilizada em diversos domínios. Há várias formas de visualização desse tipo de registro, como o gráfico de linha, gráfico de linha empilhado, dentre outros. Um problema com essas maneiras de visualizar séries temporais, contudo, é que com o aumento do número de séries, aumenta-se a dificuldade em identifica-las, distingui-las e observar padrões ou tendências. Na tentativa de mitigar esses problemas, utilizamos o conceito de profundidade de banda, que é uma noção dentro do universo de profundidade de dados funcionais que permite ordenar um conjunto desse tipo de dados com respeito à centralidade em uma distribuição. Desse modo, o presente trabalho tem como objetivo propor uma nova abordagem para a computação aproximada de profundidade de banda, que dê suporte à análise interativa de grandes coleções de séries temporais. Para atingir esses objetivos, realizamos experimentos comparando a abordagem proposta com a abordagem clássica do cálculo de profundidade de banda e com outro método de aproximação, no que diz respeito à performance e ao custo de memória adicional necessário. Assim, percebemos que os resultados obtidos mostram que o método proposto neste estudo tem uma performance melhor do que os métodos clássicos e um custo de memória consideravelmente menor do que a outra aproximação, mesmo com o aumento dos dados. Além disso, utilizamos as aproximações obtidas em uma técnica de visualização que nos permite ver a distribuição das curvas e quais delas tiveram um comportamento que pode possivelmente classificá-las como *outliers*.

Palavras-chave: Visualização de dados, Profundidade de banda, Profundidade de dados funcionais, t-digest

Abstract

Time series are one of the most common forms of data logging and are widely used in many domains. There are several ways to view this type of record, such as line chart, stacked line chart, among others. One problem with these ways of viewing time series, however, is that as the number of series increases, the difficulty in identifying, distinguishing, and observing patterns or trends increases. In an attempt to mitigate these problems, we used the concept of band depth, which is a notion within the universe of functional data depth that allows us to sort such data with respect to centrality in a distribution. Thus, this paper aims to propose a new approach to approximate band depth computation, which supports interactive analysis of large time series collections. To achieve these objectives, we performed experiments comparing the proposed approach with the classical approach of band depth calculation and another approximation method with respect to the performance and additional memory cost required. Thus, we realized that the results show that the method proposed in this study has a better performance than the classical methods and a considerably lower memory cost than the other approximation, even with the increase of data. In addition, we use the approximations obtained in a visualization technique that allows us to see the distribution of the curves and which ones had a behavior that could possibly classify them as *outliers*.

Keywords: Data visualization, Band depth, Functional data depth, t-digest

Lista de Figuras

1.1 Exemplo de *overplotting* em conjunto de dados sobre valor de ações na bolsa ao longo dos anos. Fonte: MORITZ; FISHER (2018) 14

2.1 Algumas das técnicas presentes no estudo de JAVED; MCDONNEL; ELMQVIST (2010). A primeira imagem representa um gráfico de linhas simples, em que cada linha corresponde a uma série temporal. A segunda imagem é um exemplo da técnica *small multiples*, proposto por TUFTE; GOELER; BENSON (1990), onde o gráfico de linhas tradicional é dividido em um gráfico individual para cada linha do conjunto, podendo esse ser preenchido (como na imagem) ou não. A terceira imagem corresponde à técnica de gráfico de linhas empilhado *stacked line chart*, em que, como o nome diz, as linhas que correspondem às séries temporais são empilhadas, mantendo todas as curvas no mesmo gráfico. Fonte: JAVED; MCDONNEL; ELMQVIST (2010) 18
2.2 Exemplo de gráfico utilizando técnica *DenseLines*. Fonte: MORITZ; FISHER (2018) 18

3.1 Exemplo de conjunto de dados, a região cinzenta representa a banda formada pelas curvas y_1 e y_3. Fonte: SUN; GENTON; NYCHKA (2012) 21
3.2 Exemplo de *functional boxplot*. Fonte: SUN; GENTON (2011) 27

4.1 Representação dos elementos do vetor de t-digests para um conjunto de curvas. Fonte: do autor. 31

5.1 *Boxplots* representando a distribuição dos valores de profundidade de banda obtidos por cada um dos algoritmos analisados. Em (a) vemos a distribuição das profundidades de banda originais para *dataset* de qualidade do ar (AQ), em (b) para *dataset* de corridas de táxis em 2018 (TX18), em (c) para *dataset* de corridas de táxis nos anos 2016-2018 (TX1618) e em (d) para *dataset* de corridas de táxis nos anos 2014-2018 (TX1418) Fonte: do autor. 36
5.2 *Boxplots* representando a distribuição dos valores de profundidade de banda modificada obtidos por cada um dos algoritmos analisados. Em (a) vemos a distribuição das profundidades de banda modificadas para AQ; em (b) para TX18, em (c) para TX1618 e em (d) para TX1418. Fonte: do autor. 37

5.3 Distribuição do erro relativo dos métodos de aproximação em relação à implementação *naïve* para ambas as noções de profundidade de banda. (a) Erro relativo da *Band Depth* (BD) para AQ. (b) Erro relativo da *Modified Band Depth* (MBD) para AQ. (c) Erro relativo da BD para TX18. (d) Erro relativo da MBD para TX18. (e) Erro relativo da BD para TX1618. (f) Erro relativo da MBD para TX1618. (g) Erro relativo da BD para TX1418. (g) Erro relativo da MBD para TX1418. Fonte: do autor. 38

5.4 *Functional boxplots* dos resultados dos cálculos da profundidade de banda e profundidade de banda modificada obtidos por cada método nos dias de semana em TX18. (a) *Functional boxplot* dos resultados de método *naïve* (OD). (b) *Functional boxplot* dos resultados de método *naïve* (OMD). (c) *Functional boxplot* dos resultados de aproximação utilizando matriz de *rankings* (FD). (d) *Functional boxplot* dos resultados de aproximação utilizando matriz de *rankings* (FMD). (e) *Functional boxplot* dos resultados de aproximação utilizando t-digests (TD). (f) *Functional boxplot* dos resultados de aproximação utilizando t-digests (TMD). Fonte: do autor. 40

6.1 *Functional boxplots* dos resultados do cálculo da profundidade de banda modificada obtidos pelo algoritmo proposto nos dois subconjuntos de TX18. Em (a) estão os resultados do método TMD para os dias de semana e em (b) para os finais de semana Fonte: do autor. 46

Lista de Tabelas

5.1 Tempo levado por cada um dos algoritmos para calcular as profundidades de todas as curvas. Fonte: do autor. 41
5.2 Tempo médio para calcular a profundidade de banda de uma curva em cada um dos algoritmos. Fonte: do autor. 41
5.3 Tempo levado por cada um dos algoritmos para calcular as profundidades modificadas de todas as curvas. Fonte: do autor. 42
5.4 Tempo médio para calcular a profundidade de banda modificada de uma curva em cada um dos algoritmos. Fonte: do autor. 42
5.5 Tempo para construção de estruturas de dados necessárias para o cálculo dos métodos de aproximação de profundidade de banda. Fonte: do autor. 42
5.6 Custo de memória adicional - para os métodos que a utilizam - em cada um dos conjuntos de dados analisados. Fonte: do autor. 43

Lista de Acrônimos

IQR	*Inter-Quartile Range*
PDF	Profundidade de Dados Funcionais
BD	*Band Depth*
MBD	*Modified Band Depth*
AQ	*dataset* de qualidade do ar
TX18	*dataset* de corridas de táxis em 2018
TX1618	*dataset* de corridas de táxis nos anos 2016-2018
TX1418	*dataset* de corridas de táxis nos anos 2014-2018
OD	método *naïve*
FD	aproximação utilizando matriz de *rankings*
TD	aproximação utilizando t-digests
OMD	método *naïve*
FMD	aproximação utilizando matriz de *rankings*
TMD	aproximação utilizando t-digests
CDF	*cumulative distribution function*

Lista de Algoritmos

1 Algoritmo para calcular as profundidades de bandas de um conjunto de curvas. 22
2 Algoritmo para checar se uma curva está completamente contida na banda definida por duas curvas. 22
3 Algoritmo para medir a quantos pontos de uma curva estão dentro de uma banda. 24
4 Algoritmo para calcular as profundidades de bandas modificadas de um conjunto de curvas. 24
5 Algoritmo para achar *rankings* máximo e mínimo das curvas utilizando T . . . 31
6 Algoritmo para achar proporções das curvas de um conjunto usando T 32

Sumário

1 Introdução **13**

2 Trabalhos Relacionados **16**
2.1 Computação Aproximada da Profundidade de Banda 16
2.2 Análise Visual de Dados Temporais 17

3 Background **20**
3.1 Profundidade de Banda 20
3.2 Profundidade de Banda Modificada 23
3.3 Cálculo da Profundidade de Banda Utilizando Matriz de *Rankings* 25
3.4 *Functional Boxplot* 27
3.5 T-Digest 28

4 Método de aproximação proposto **30**

5 Análise experimental **33**
5.1 Metodologia 33
5.1.1 Os Dados 33
5.1.2 Comparação dos Métodos de Cálculo de Profundidade de Banda 34
5.1.3 Comparação do Custo Computacional das Implementações dos Métodos de Cálculo de Profundidade de Banda 34
5.1.3.1 Medindo o Desempenho de Tempo dos Algoritmos 34
5.1.3.2 Medindo o Custo de Memória dos Algoritmos 35
5.2 Resultados Experimentais 35
5.2.1 Comparação dos Resultados dos Algoritmos 36
5.2.2 Comparação do Desempenho dos Algoritmos 41
5.2.3 Comparação do Custo de Memória dos Algoritmos 43
5.3 Discussão 43

6 Casos de Uso **45**

7 Conclusão **48**

Referências **50**

1

Introdução

Uma das formas mais comuns de se registrar dados consiste em utilizar séries temporais, em que valores mudam continuamente ao longo do tempo, porém devem ser medidos e amostrados em intervalos discretos. Esse tipo de registro de dados é amplamente utilizado por analistas em diversos domínios como finanças, dados climáticos, entre outros, como visto em FULCHER; LITTLE; JONES (2013).

Para visualizar múltiplas séries temporais geralmente são utilizados gráficos de linha onde cada linha representa uma série, demonstrado em PLAYFAIR (1801); AIGNER et al. (2011). Entretanto, a medida que o número de linhas no gráfico aumenta, fica cada vez mais difícil de conseguir identificar padrões ou tendências no conjunto de dados devido ao problema conhecido como *overplotting*, que nada mais é que a sobreposição das linhas devido à grande quantidade e pode ser observado na Figura 1.1. Para tentar auxiliar na resolução do problema mencionado acima, pesquisas têm sido feitas para utilizar o conceito de densidade em determinadas áreas do gráfico como ferramenta de visualização, permitindo ver áreas com maior ou menor concentração de linhas, o que pode ser observado no trabalho de MORITZ; FISHER (2018). Apesar disso, a visualização baseada em densidade se restringe a mostrar regiões com mais ou menos curvas no gráfico, não sendo possível identificar, por exemplo, comportamentos anômalos em regiões de maior densidade ou padrões em regiões de densidade menor. Por este motivo, existe uma necessidade de uma outra abordagem para a visualização de gráficos de linha com muitas séries temporais que seja eficiente e robusta para cobrir tanto os casos tradicionais como outras situações que não são contempladas atualmente.

O conceito de Profundidade de Dados Funcionais (PDF) (*functional data depth*) é uma importante ferramenta para análise de dados funcionais. Ele permite ordenar um conjunto dese tipo de dados com respeito a centralidade em uma distribuição. Dentre as noções de PDF uma das mais tradicionais e importantes é a profundidade de banda - *Band Depth* (BD), em inglês - proposta por LÓPEZ-PINTADO; ROMO (2009). Apesar de sua importância para a análise de dados funcionais, a computação da profundidade de banda apresenta um desafio para a sua aplicação em sistemas de visualização interativa de dados.

Uma maneira ingênua para calcular a profundidade de banda dado um conjunto de

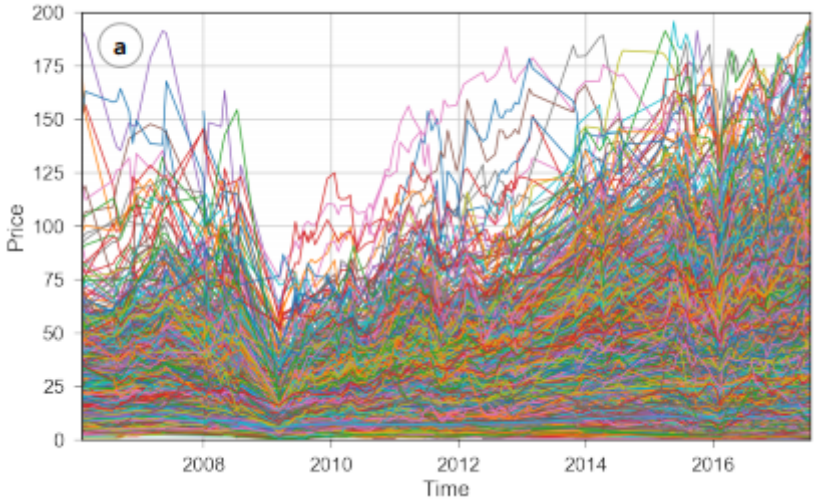


Figura 1.1: Exemplo de *overplotting* em conjunto de dados sobre valor de ações na bolsa ao longo dos anos. Fonte: MORITZ; FISHER (2018)

funções/curvas (que, no nosso caso, são as linhas do gráfico de linhas que representam as séries temporais) a partir da definção é checar para uma dada curva, dentro de quantas bandas definidas por pares de curvas ela se encontra. Este algoritmo tem uma complexidade de tempo de ordem cúbica, aquém do ideal para coleções de dados que tenham muitas curvas. Uma alternativa proposta por SUN; GENTON; NYCHKA (2012) consiste em construir uma matriz de rankings como uma etapa de pré-processamento. O uso desta matriz acelera o tempo de execução, pois o cálculo da profundidade para cada curva ocorre em tempo linear sobre o número de pontos da curva; ainda assim, o fato dessa abordagem depender da construção de uma matriz ocasiona um uso exagerado de memória para coleções de dados maiores.

Diversos estudos já demonstraram a importância de otimizar o tempo de execução e memória (LIU; HEER (2014); LARA PAHINS; FERREIRA; COMBA (2019)) em sistemas de visualização interativa de dados. O trabalho de LIU; HEER (2014), descreve como a latência afeta negativamente o uso de sistemas de análise visual exploratória, mostrando que um delay de 500 milissegundos já tem um impacto considerável na análise visual, reduzindo interação e cobertura dos dados, bem como o número de observações, generalizações e formulação de hipóteses, entre outros efeitos indesejados para um sistema de visualização. A necessidade de otimização do uso de memória decorre do fato de que, para grandes coleções de dados, a utilização de estruturas de dados ou abordagens que não levem em consideração essa restrição pode resultar em problemas de execução.

O objetivo deste trabalho é propor uma nova abordagem para a computação (aproximada) de profundidade de banda que dê suporte a análise interativa de grandes coleções de séries temporais. Para isso pretendemos fazer uma comparação dos métodos citados anteriormente, bem como propor da utilização de uma estrutura de dados baseada no uso do *quantile data sketch t-digest* inspirados pelo trabalho de LARA PAHINS; FERREIRA; COMBA (2019).

Quantile data sketches são estruturas de dados que aproximam a distribuição de dados univariados de forma que "pode ser facilmente atualizada com dados novos ou modificados e suporta um conjunto de consultas cujos resultados aproximam as consultas no conjunto de dados completo", dito no trabalho de PHILLIPS (2016). Através do uso do *t-digest* iremos de forma semelhante ao trabalho proposto por SUN; GENTON; NYCHKA (2012), mas eliminando a limitação de memória utilizada, bem como possibilitando operações de filtragem comuns em sistemas de visualização de dados. Finalmente, iremos demonstrar como esse conceito pode ser utilizado para a visualização de gráficos de linhas que representam um grande número de séries temporais.

2

Trabalhos Relacionados

Neste Capítulo apresentaremos uma revisão de artigos relacionados aos temas abordados em nosso trabalho.

2.1 Computação Aproximada da Profundidade de Banda

A teoria proposta por LÓPEZ-PINTADO; ROMO (2009) introduz uma das noções mais clássicas de profundidade de dados funcionais: a profundidade da banda (*band depth*). Nesse trabalho essa função é definida através de uma abordagem baseada em gráficos. Seja x uma curva, n o número de curvas e j um número tal que $2 \leq j \leq n$, a profundidade de banda de x corresponde à proporção de bandas delimitadas por j curvas que contém completamente o gráfico de x. Quando as curvas são muito irregulares poucas bandas irão conter completamente uma curva, resultando em muitas curvas com o mesmo valor de profundidade e, consequentemente, um *ranking* não muito útil. Para resolver isso, LÓPEZ-PINTADO; ROMO (2009) propuseram em seu trabalho, além da noção original, uma versão modificada - *Modified Band Depth* (MBD), em inglês - da profundidade de banda que mede a proporção de tempo que uma curva x está dentro de cada banda. Essas noções são de grande importância pois o trabalho é baseado nas mesmas, entretanto a forma original de computá-las se mostra muito ineficiente para grandes coleções de dados. Mais detalhes do cálculo, da computação e do custo computacional das noções de profundidade de banda e profundidade de banda modificada, bem como uma utilização dessas noções em uma técnica de visualização serão abordados no Capítulo 3.

O estudo de SUN; GENTON; NYCHKA (2012) apresenta uma forma de aproximar o cálculo da profundidade de banda proposta por LÓPEZ-PINTADO; ROMO (2009) através da construção de uma matriz de rankings como uma etapa de pré-processamento. Essa matriz é uma matriz p x n, onde n é o número de curvas sendo avaliadas e p o número de pontos de cada curva, dessa forma cada elemento da matriz corresponde ao *ranking* da curva n no ponto p, cada coluna representa uma curva e cada linha um *data point*. Dada uma curva, esta matriz pode ser usada para calcular de maneira eficiente quantas curvas estão sempre acima ou sempre abaixo dela e assim calcular a profundidade de banda. Isto acelera o desempenho de tempo para obter

os valores de profundidade de banda, pois o cálculo se utiliza apenas dos *rankings* máximo e mínimo da curva, obtidos em um tempo da ordem de $O(p)$. Entretanto, devido à construção da matriz de rankings, a memória necessária para essa abordagem pode chegar a valores muito grandes caso a coleção de dados tenha muitas observações.

2.2 Análise Visual de Dados Temporais

A visualização de séries temporais é um tema em aberto na comunidade de pesquisa de visualização de dados. Em decorrência desse fato existem diversos métodos para a visualização dessas séries, o estudo de JAVED; MCDONNEL; ELMQVIST (2010) mostra diversas dessas técnicas e também realiza um experimento onde eles classificam os métodos listados de acordo com diversos critérios e conduzem um estudo com usuários para entender em que situações determinados métodos se sobrepõem aos outros. Na Figura 2.1 podemos observar algumas das técnicas mais clássicas para a visualização de muitas séries temporais que são contempladas no trabalho proposto. Um dos problemas mais comuns de acontecer nesse tipo de visualização, principalmente nas técnicas mais tradicionais, e que é levado em consideração no estudo é o *overplotting*, a sobreposição de linhas. Um exemplo desse problema pode ser visto, novamente, na Figura 1.1.

Uma das opções para solucionar o *overplotting* consiste no uso de agregações, o artigo proposto por MORITZ; FISHER (2018) faz uso desse conceito para apresentar uma nova técnica para calcular e mostrar uma representação discreta da densidade de séries temporais, chamada *DenseLines*, com o objetivo de permitir a visualização, e posterior análise, de muitas séries temporais de forma escalável. O *core* da técnica apresentada é computar a densidade como o número de séries temporais que que passam por um espaço particular nas dimensões de tempo e do valor a ser analisado. Para permitir esse cálculo é necessário realizar a normalização da contribuição da densidade de cada linha pelo tamanho do seu arco, de modo que cada série tem o mesmo peso total. Em resumo, a técnica consiste de computar a densidade dividindo o gráfico em regiões e a densidade de cada região mede o número de linhas que passam pela região, resultando em um gráfico semelhante a um *heatmap*, como podemos observar na Figura 2.2. Apesar de ser uma técnica interessante, possui algumas limitações, dentre elas: a técnica não permite a observação de linhas individuais, fazendo com que seja impossível observar comportamentos incomuns em regiões mais comuns, de modo que apenas outliers relativamente óbvios sejam possíveis de ser visualizados.

No artigo proposto por MIRANDA et al. (2018) os autores propõem uma nova estrutura de dados chamada *Time Lattice*, que apresenta um *overhead* de memória menor e que faz uso da hierarquia temporal implícita para permitir *queries* OLAP (*online analytical processing*), que são consultas que incluem restrições e *group-by*'s e são de fundamental importância para explorar e comparar visualmente séries temporais em diferentes escalas. A estrutura proposta é derivada do conceito de cubo de dados (GRAY et al. (1997)), que é um método projetado para

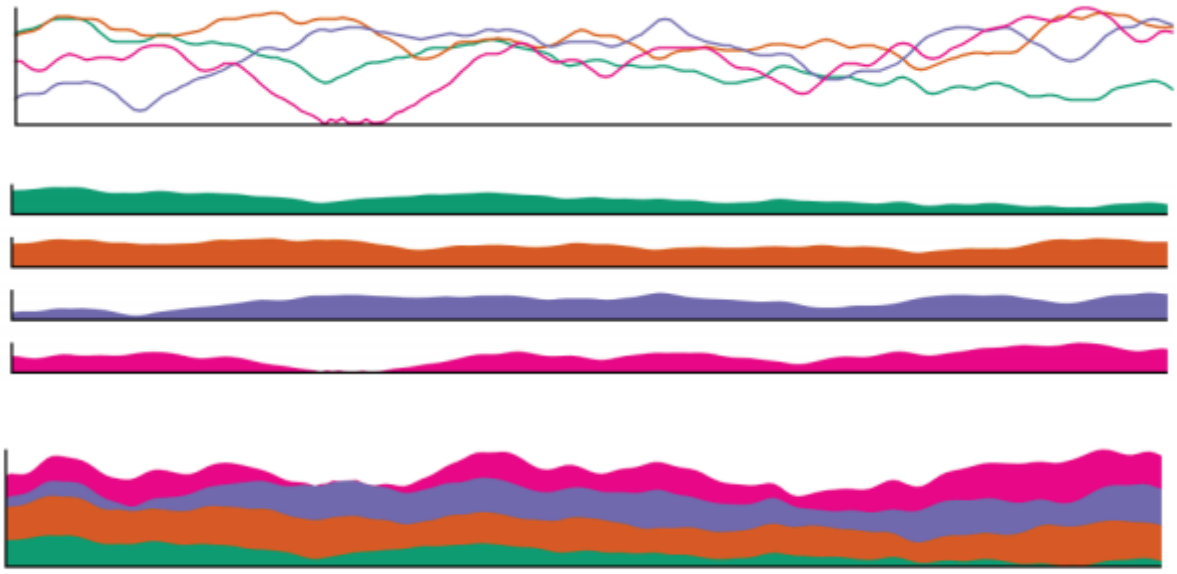


Figura 2.1: Algumas das técnicas presentes no estudo de JAVED; MCDONNEL; ELMQVIST (2010). A primeira imagem representa um gráfico de linhas simples, em que cada linha corresponde a uma série temporal. A segunda imagem é um exemplo da técnica *small multiples*, proposto por TUFTE; GOELER; BENSON (1990), onde o gráfico de linhas tradicional é dividido em um gráfico individual para cada linha do conjunto, podendo esse ser preenchido (como na imagem) ou não. A terceira imagem corresponde à técnica de gráfico de linhas empilhado *stacked line chart*, em que, como o nome diz, as linhas que correspondem às séries temporais são empilhadas, mantendo todas as curvas no mesmo gráfico. Fonte: JAVED; MCDONNEL; ELMQVIST (2010)

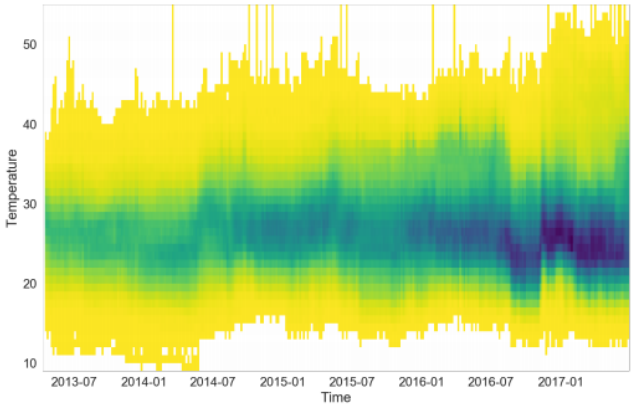


Figura 2.2: Exemplo de gráfico utilizando técnica *DenseLines*. Fonte: MORITZ; FISHER (2018)

responder consultas de agregações eficientemente. No caso de séries temporais, cada dimensão de tempo é modelada como uma dimensão do cubo de dados. A propriedade de hierarquia temporal implícita decorre do fato das medidas de tempo correspondentes a cada uma das dimensões temporais serem hierarquicamente dependentes. O *Time Lattice* computa, então, apenas um subconjunto do cubo de dados original, se aproveitando da propriedade citada acima para computar eficientemente resultados de consultas à medida que elas aparecem. Esse trabalho se mostra como uma alternativa aos métodos tradicionais de cubos de dados que, apesar de suportar consultas OLAP e interativas, não são adequados para lidar com coleções grandes de dados, visto que a exigência de memória cresce exponencialmente com a quantidade de dados. Além disso, nenhuma das recentes estruturas de cubos com maior eficiência espacial permite atualizações. Dessa forma, o cubo precisa ser re-computado todas as vezes que chegam novas observações, sendo impraticável sua utilização em serviços que possuem *streaming* de dados.

Apesar da importância das agregações, esse conceito pode esconder padrões importantes presentes na distribuição dos dados. Os trabalhos citados a seguir propõem novos métodos que buscam resolver esse problema:

No trabalho de DORAISWAMY et al. (2014) os autores propõem uma técnica que auxilia na exploração orientada a eventos de grandes conjuntos de dados espaço-temporais. Neste trabalho, os dados são modelados como funções escalares variáveis no tempo e fizeram uso de topologia computacional para automaticamente identificar eventos em diferentes *data slices*, que são diferentes porções espaciais/ou temporais dos dados.O método proposto nesse trabalho é capaz de lidar com uma grande quantidade de dados por meio de uma abordagem baseada em agrupamento e indexação dos eventos, facilitando, assim, a exploração interativa dos padrões de eventos em tempo real.

O estudo de LARA PAHINS; FERREIRA; COMBA (2019) apresenta a estrutura de dados *Quantile Datacube Structure* (QDS), com o objetivo de resolver a limitação das abordagens baseadas em cubos de dados que, apesar de robustas, não levam em consideração que as agregações utilizadas para produzir os cubos não representam a distribuição dos dados que estão sendo analisados e também possuem um *overhead* de memória elevado. Para isso, a estrutura QDS se utiliza de um esquema de aproximação de distribuição não-paramétrico, chamado *p-digest*, baseado no *quantile sketch t-digest* (DUNNING; ERTL (2019)), e também de um novo esquema de indexação de cubo de dados que reduz a utilização de memória quando comparado à métodos de cubos clássicos. Isso permite a filtragem interativa dos dados, fundamental na tarefa de visualização exploratória interativa de grandes coleções de dados espaço-temporais, ao mesmo tempo que aproxima precisamente a distribuição de variáveis quantitativas de interesse.

Como dito no Capítulo 1, nosso trabalho propõe uma nova abordagem para a computação aproximada de profundidade de banda que dê suporte a análise interativa de grandes coleções de séries temporais, buscando resolver os problemas observados nos trabalhos citados acima.

3

Background

O presente capítulo fornece mais detalhes a respeito da definição das diferentes noções de profundidade de banda abordadas por este trabalho, bem como sobre os algoritmos de cálculo propostos na literatura para o cáculo dessas profundidades.

3.1 Profundidade de Banda

LÓPEZ-PINTADO; ROMO (2009) introduziram a noção de profundidade de banda através de uma abordagem baseada em gráficos. Essa definição é uma das noções mais clássicas no universo de PDF, permitindo ordenar um conjunto de curvas com respeito a centralidade e, assim, gerar um *ranking* das curvas mais centrais até as mais marginais. O gráfico de uma função $x(t)$ é o subconjunto do plano $G(x) = \{(t, x(t)) : t \in I'\}$ onde I' é um intervalo no conjunto dos números reais. Para cada função x em $x_1, ..., x_n$ onde n corresponde ao número de observações (curvas) e o valor j, onde $2 \leq j \leq n$,

$$BD_n^{(j)}(x) = \binom{n}{j}^{-1} \sum_{1 \leq i_1 < i_2 < ... < i_j \leq n} I\{G(x) \subseteq B(x_{i_1}, x_{i_2} ..., x_{i_n})\} \tag{3.1}$$

expressa a proporção de bandas $B(x_{i_1}, ..., x_{i_j})$ determinadas por j curvas diferentes $x_{i_1}, ..., x_{i_j}$ que contém completamente o gráfico de x, essa situação pode ser observada na Figura 3.1, em que a curva y_2 está completamente contida na banda formada pelas curvas y_1 e y_3. Na Equação 3.1, I denota a função indicadora, ou seja, a função que indica se o elemento pertence ao conjunto, assumindo neste caso o valor 1, e 0 em caso contrário. Dessa forma, seja J um valor tal que $2 \leq J \leq n$, para as funções $x_1, x_2, ..., x_n$, a profundidade de banda para qualquer uma dessas curvas é dada pela Equação (3.2).

$$BD_{n,J}(x) = \sum_{j=2}^{J} BD_n^{(j)}(x) \tag{3.2}$$

Um dos problemas que existe com essa definição de profundidadede banda é que ela é muito sensível a ruídos nos dados. De fato, uma pequena alteração em um ponto de uma dada

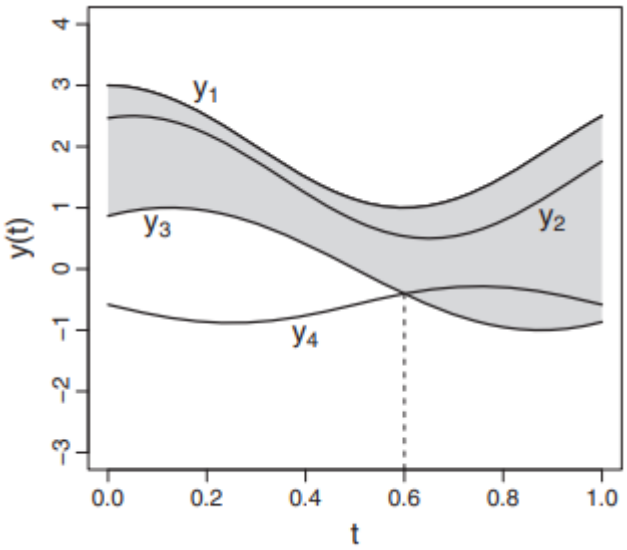


Figura 3.1: Exemplo de conjunto de dados, a região cinzenta representa a banda formada pelas curvas y_1 e y_3. Fonte: SUN; GENTON; NYCHKA (2012)

curva, pode fazer com que ela não esteja contida em nenhuma banda que não tenha a curva dada na sua fronteira. Além disso, quando há o grande número de curvas e estas são irregulares, ou seja, se cruzam em um ou mais pontos como é o caso da curva y_4 na Figura 3.1 que pertence apenas parcialmente à banda ilustrada, o número de bandas que irão conter completamente uma curva é muito pequeno e isso resulta em muitas curvas com o mesmo valor de profundidade. Por sua vez, a consequência da existência de muitas curvas com um mesmo valor de profundidade é que o *ranking* das curvas gerado fica fracamente definido.

Apresentamos a seguir os algoritmos *naïve* para cálculo de profundidade de banda: o Algoritmo 1 mostra como é feito o cálculo da profundidade de cada uma das curvas de uma coleção de dados, e o Algoritmo 2 mostra como é verificado se o gráfico de uma curva está completamente contido dentro da banda delimitada por duas curvas, sendo usando como sub-rotina no Algoritmo 1.

No Algoritmo 1, que é a implementação direta da definição dada por LÓPEZ-PINTADO; ROMO (2009), é possível observar a presença de três *for loops* aninhados: os dois mais internos são responsáveis por gerar os pares de curvas que delimitam as bandas e o mais externo itera sobre cada uma das curvas, para verificar se a mesma está completamente contida na banda definida pelas curvas definidas pelos dois *for* internos. O custo computacional para a execução desse algoritmo é da ordem de $O(n^3 p)$, onde n é o número de curvas e p o número de pontos por curva.

Algoritmo 1 Algoritmo para calcular as profundidades de bandas de um conjunto de curvas.

Input: Um array de curvas C com n curvas de tamanho p
  for $i = 0$ to $n-1$ **do**
    $C[i].bandDepth = 0.0$
  end for
  for $k = 0$ to $n-1$ **do**
    for $i = 0$ to $n-2$ **do**
      for $j = i+1$ to $n-1$ **do**
        $C[k].bandDepth \leftarrow C[k].bandDepth + check_curve(C[k], C[i], C[j])$
      end for
    end for
  end for
  $nchoose2 \leftarrow \binom{n}{2}$
  for $i = 0$ to $n-1$ **do**
    $C[i].bandDepth \leftarrow C[i].bandDepth/nchoose2$
  end for

Algoritmo 2 Algoritmo para checar se uma curva está completamente contida na banda definida por duas curvas.

Input: Três curvas c_1, c_2, c_3 de tamanho p
Output: 1 se $c_1 \subseteq B(c_2, c_3)$, 0 caso contrário
  for $i = 0$ to $p-1$ **do**
    $a \leftarrow c1.values[i]$
    $b \leftarrow min(c2.values[i], c3, values[i])$
    $c \leftarrow max(c2.values[i], c3, values[i])$
    if $(a < b \vee a > c)$ **then**
      return 0
    end if
  end for
  return 1

3.2 Profundidade de Banda Modificada

No mesmo trabalho, LÓPEZ-PINTADO; ROMO (2009) também introduziram uma outra noção de profundidade de banda que, em vez de utilizar a função indicadora utiliza uma definição mais flexível medindo o conjunto em que a função está dentro da banda correspondente, dando origem a profundidade de banda modificada. Para qualquer função x em $x_1,...,x_n$ e para $2 \leq j \leq n$, seja

$$A_j(x) \equiv A(x;x_{i_1},...,x_{i_j}) \equiv \{t \in I : min_{r=i_1,...,i_j} x_r(t) \leq x(t) \leq max_{r=i_1,...,i_j} x_r(t)\} \tag{3.3}$$

o subconjunto do intervalo I onde a função x está na banda determinada pelas observações $x_{i_1},...,x_{i_j}$. Seja λ a medida de Lebesgue nos números reais, $\lambda_r(A_j(x)) = \lambda(A_j(x))/\lambda(I)$ nos dá a proporção do domínio em que x está na banda, logo:

$$MBD_n^{(j)} = \binom{n}{j}^{-1} \sum_{1 \leq i_1 < i_2 < ... < i_j \leq n} \lambda_r(A(x;x_{i_1},...,x_{i_j})), 2 \leq j \leq n \tag{3.4}$$

é uma versão mais flexível da Equação 3.1: se x estiver sempre dentro da banda, o valor $\lambda_r(A_j(x))$ será 1 da mesma forma que seria na função indicadora na noção anterior de profundidade de banda. Da mesma maneira que a Equação (3.2), para um valor J onde $2 \leq J \leq n$ e um conjunto de funções $x_1, x_2, ..., x_n$ a profundidade de banda modificada de qualquer uma dessas funções é dada pela Equação (3.5).

$$MBD_{n,J}(x) = \sum_{j=2}^{J} MBD_n^{(j)}(x) \tag{3.5}$$

Essa definição se mostra mais robusta à presença de ruídos no conjuntos de dados, situação em que a noção original de banda não tem um bom desempenho por dar o mesmo valor para um grande número de curvas.

Da mesma forma que na Seção 3.1, mostraremos abaixo os algoritmos *naïve*, que são a implementação direta das definições, responsáveis por calcular quantos pontos de uma curva estão dentro de uma banda (Algoritmo 3) e também a profundidade de banda modificada de cada uma das curvas de um conjunto de séries temporais (Algoritmo 4).

Da mesma forma que o Algoritmo 1, o Algoritmo 4 também se utiliza de três *for loops* aninhados, dois para gerar os pares de curvas e um para iterar sobre as curvas, além do algoritmo de contagem dos pontos da curva que estão dentro da banda. Isso resulta no mesmo custo computacional de $O(n^3 p)$.

Algoritmo 3 Algoritmo para medir a quantos pontos de uma curva estão dentro de uma banda.

Input: Três curvas c_1, c_2, c_3 de tamanho p
Output: Proporção de c_1 que está dentro de $B(c_2, c_3)$

```
for i = 0 to p − 1 do
    count ← 0
    a ← c1.values[i]
    b ← min(c2.values[i], c3, values[i])
    c ← max(c2.values[i], c3, values[i])
    if (b <= a ∧ a <= c) then
        count ← count + 1
    end if
end for
return count
```

Algoritmo 4 Algoritmo para calcular as profundidades de bandas modificadas de um conjunto de curvas.

Input: Um conjunto de curvas C com n curvas de tamanho p

```
for i = 0 to n − 1 do
    C[i].modifiedBandDepth = 0.0
end for
for k = 0 to n − 1 do
    for i = 0 to n − 2 do
        for j = i + 1 to n − 1 do
            proportion ← count_points_between(C[k], C[i], C[j])/p
            C[k].modifiedBandDepth ← C[k].modifiedBandDepth + proportion
        end for
    end for
end for
nchoose2 ← (n 2)
for i = 0 to n − 1 do
    C[i].modifiedBandDepth ← C[i].bandDepth/nchoose2
end for
```

3.3 Cálculo da Profundidade de Banda Utilizando Matriz de *Rankings*

A elevada complexidade computacional dos algoritmos *naïve* para o calculo da profundidade de banda impede sua aplicação em cenários com um elevado número de curvas e/ou onde a latencia do cálculo é restrita como no caso de visualizações interativas (LIU; HEER (2014)). Este fato evidencia a necessidade de algoritmos mais eficientes para o cálculo da profundidade de banda. Uma alternativa proposta por SUN; GENTON; NYCHKA (2012) é a utilização de uma matriz de *rankings* pré-calculada para acelerar o cálculo do número de bandas que contém uma curva dada. Para tanto, os autores propõem a aproximação da banda original dada na Equação (3.1) por

$$BD_n^{(2)}(x) = \binom{n}{2}^{-1} \sum_{1 \le i_1 < i_2 \le n} I\{G(x) \subseteq B(x_{i_1}, x_{i_2})\}, \tag{3.6}$$

e da banda modificada da Equação (3.4):

$$MBD_n^{(2)}(x) = \binom{n}{2}^{-1} \sum_{1 \le i_1 < i_2 \le n} \lambda_r(A(x; x_{i_1}, x_{i_2})) \tag{3.7}$$

Seja M a matriz de dados $p \times n$ representando n curvas contendo p pontos. Para construir a matriz de rankings, primeiro ordenamos os n *data points* de cada linha da matriz do menor para o maior e em seguida salvamos os *rankings* $r_{ij} \in \{1, ..., n\}$ na matriz de *ranking* $R_{p \times n}$. Como podemos ver no exemplo a seguir:

$$M = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{array} \begin{array}{c} \begin{array}{ccc} f_1 & f_2 & f_3 \end{array} \\ \begin{pmatrix} 4 & 9 & 3 \\ 11 & 2 & 7 \\ 6 & 8 & 10 \\ 12 & 5 & 1 \end{pmatrix} \end{array} \rightarrow M_{ordered} = \begin{pmatrix} (3, f_3) & (4, f_1) & (9, f_2) \\ (2, f_2) & (7, f_3) & (11, f_1) \\ (6, f_1) & (8, f_2) & (10, f_3) \\ (1, f_3) & (5, f_2) & (12, f_1 \end{pmatrix} \rightarrow R = \begin{pmatrix} 2 & 3 & 1 \\ 3 & 1 & 2 \\ 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$$

Com a matriz R construída, para a j-ésima curva:

$$n_b = \min_{1 \le i \le p} (r_{ij}) - 1 \tag{3.8}$$

representa quantas curvas estão sempre abaixo e

$$n_a = n - \max_{1 \le i \le p} (r_{ij}) \tag{3.9}$$

representa quantas curvas estão sempre acima. Com esses valores é possível obter o valor de profundidade a partir de:

$$BD(x_j) = \binom{n}{2}^{-1} n_a n_b + (n-1) \tag{3.10}$$

na Equação acima, $n-1$ representa o número de bandas que tem x_j na borda. Da mesma forma que a profundidade de banda original, a profundidade de banda modificada também pode ser calculada usando a abordagem proposta por SUN; GENTON; NYCHKA (2012), modificando as Equaçoes (3.8) e (3.9), da seguinte forma

$$n_b(i) = r_{ij} - 1 \tag{3.11}$$

$$n_a(i) = n - r_{ij} \tag{3.12}$$

obtemos, em determinado ponto i, quantas curvas estão acima e quantas curvas estão abaixo de x_j.

$$MBD(x_j) = \binom{n}{2}^{-1} \left(\frac{\sum_{1 \leq i \leq p} n_a(i) n_b(i)}{p} + (n-1) \right) \tag{3.13}$$

A Equação acima nos dá, então, a profundidade de banda modificada para a curva j. O termo $n_a(i)n_b(i)$ representa, no ponto i, a quantidade de bandas que contêm a curva, dessa forma o termo $\frac{\sum_{1 \leq i \leq p} n_a(i) n_b(i)}{p}$ nos dá a proporção de tempo que a curva está contida nas bandas que não contêm ela na fronteira, enquanto $(n-1)$ denota o número de bandas em que a curva está contida na fronteira.

O custo computacional do método para calcular a profundidade de banda pode ser dividido em duas partes, o custo para construir a matriz de *rankings* e o custo de achar a profundidade de banda. O custo para a construção da matriz é dependente do algoritmo utilizado para ordenar as linhas - que representam cada instante de tempo - da matriz de valores inicial, $O(pnlog(n))$ no caso do *quick sort*. Já o custo para calcular a profundidade de banda de uma curva consiste em achar os *rankings* máximo e mínimo da curva, da ordem de $0(p)$ e fazer as operações necessárias com esses *rankings*, logo, para um conjunto com n curvas, o custo é da ordem de $O(np)$.

Comparado com o custo para cálculo de profundidade de banda do método *naïve* original, o ganho em desempenho atingido por esse método proposto é notável. Apesar disso, uma desvantagem desse método é a necessidade de memória adicional para o cálculo da profundidade de banda, dada a necessidade de construir a matriz de *rankings* previamente, acarretando em um *overhead* de memória da ordem de $O(np)$, que pode ser significativo para conjuntos com muitas curvas ou com muitos pontos por curva. Outra desvantagem que é importante mencionar é que, apesar de o artigo original de SUN; GENTON; NYCHKA (2012) afirmar que o método realiza o cálculo exato da profundidade de banda, essa abordagem não dá resultados exatos em alguns casos, como por exemplo quando há valores repetidos no conjunto de dados.

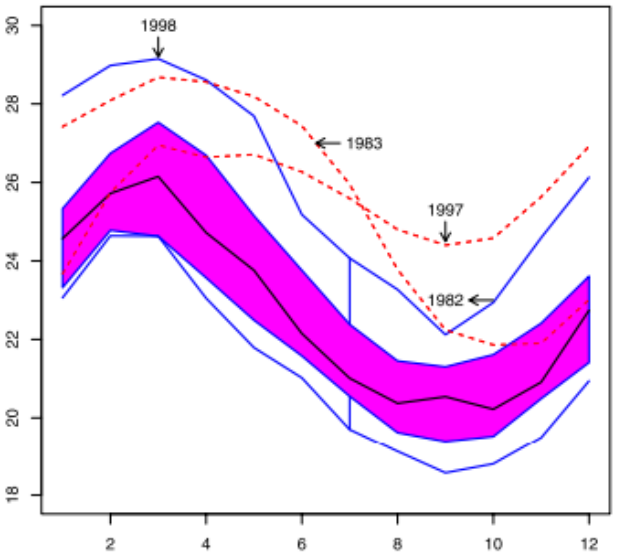


Figura 3.2: Exemplo de *functional boxplot*. Fonte: SUN; GENTON (2011)

3.4 *Functional Boxplot*

O trabalho de SUN; GENTON (2011) propõe uma ferramenta exploratória para visualizar dados funcionais: o *functional boxplot* e também sua generalização, o *enhanced functional boxplot*, a partir do ordenamento de centralidade introduzido pela noção de profundidade de banda para dados funcionais.

De forma semelhante à sua versão original, o *functional boxplot* mostra: o envelope da região 50% central (utilizando o conceito de profundidade de banda, 50% dos dados que têm as maiores profundidades) que corresponde ao *Inter-Quartile Range* (IQR), representado pela caixa (*box*) no *boxplot*; a curva mediana que representa a curva que tem a maior profundidade de banda, observada como a marca central das caixas no *boxplot* clássico; e o envelope máximo sem *outliers*, esse envelope é obtido utilizando a regra empírica de 1.5 vezes o IQR, ou seja, inflando o envelope da região 50% central por 1.5 vezes a amplitude da mesma, dessa forma qualquer curva fora desse envelope externo é considerada como um *outlier* em potencial, esse envelope é mostrado como linhas saindo das extremidades das caixas na versão tradicional. Na Figura 3.2 podemos observar um exemplo de *functional boxplot* onde a curva preta representa a curva mediana, a região rosa representa a região 50% central, as curvas azuis correspondem ao envelope máximo sem *outliers* enquanto que as curvas que estão tracejadas são consideradas *outliers* em potencial por estarem passando por fora do envelope anterior. A escolha de mostrar os 50% centrais dos dados no *functional boxplot* é justificada pois eles dão uma boa indicação da distribuição das curvas ao mesmo tempo que são robustos em relação a *outliers* ou valores extremos

3.5 T-Digest

Com o objetivo de diminuir o custo de memória adicional apresentado na abordagem de SUN; GENTON; NYCHKA (2012), optamos por usar outra estrutura de dados para o cálculo da profundidade de banda, o t-digest.

O t-digest foi proposto no trabalho de DUNNING; ERTL (2019) para computar aproximações de estatísticas de ordem com alta precisão e de maneira *on-line*, ou seja, sem as necessidades de reter todas as amostras ou aguardar até que todos os dados tenham sido processados, problemas encontrados nos algoritmos clássicos para resolver esse problema. A computação desse tipo de estatística nos métodos tradicionais geralmente é feita através da ordenação dos dados e em seguida obtenção dos quantis através de interpolação ou do re-processamento de todas as amostras dentro de intervalos quantílicos específicos. Essas abordagens são impraticáveis para conjuntos de dados muito grandes ou quando há a necessidade de calcular os quantis de muitos subconjuntos, aumentando o interesse em métodos de aproximação *on-line*.

O t-digest sumariza a Função de Distribuição Acumulada - *cumulative distribution function* (CDF) - de uma variável aleatória, definida pela equação

$$F_X(x) = P(X \leq x) \tag{3.14}$$

através de um conjunto de valores ponderados chamados de centróides controlados por um parâmetro de compressão δ, que controla o número máximo de centróides que serão obtidos. O lado direito da (3.14) representa a probabilidade p da variável aleatória X ter um valor menor ou igual a x. Além dessa função, o t-digest também é capaz de sumarizar a função inversa da CDF, a função Quantil mostrada na Equação (3.15)

$$F_X^{-1}(p) = x \tag{3.15}$$

A partir da distribuição de probabilidade de uma variável aleatória, a função Quantil especifica o valor x da variável tal que a probabilidade da variável ser menor ou igual a esse valor é igual à probabilidade fornecida p.

Quando essa estrutura recebe um conjunto de dados, a definição dos centróides é feita através do agrupamento dos elementos em subsequências de tamanhos variados, de acordo com uma estratégia adaptativa. Essa estratégia busca dar prioridade para quantis nas extremidades da distribuição dos dados, ou seja, o tamanho das subsequências para centróides próximos do início ou do fim do conjunto de dados é menor do que na região central, tornando a estimativa para quantis extremos mais precisa do que para regiões próximas da mediana, com o objetivo de melhorar a detecção de *outliers*.

O processo de construção de um t-digest é feito a partir de um vetor de valores, que são os elementos do conjunto passado para a estrutura. A partir desses vetores são encontrados os centróides utilizando a estratégia mencionada no parágrafo anterior.

A estimativa da função Quantil fornecida pelo t-digest segue o seguinte processo: a partir do quantil q desejado, o peso de cada centróide é dividido em duas partes iguais à esquerda e à direita do centróide. Então lista ordenada de centróides é percorrida, acumulando os pesos já vistos e comparando-os com $q|D|$, onde $|D|$ corresponde ao tamanho do conjunto de dados, que também é igual à soma dos pesos dos centróides. Se a acumulação alcançar o valor $q|D|$ em um centróide, o valor desse centróide será retornado. Por outro lado, se a correspondência acontecer entre dois centróides, o valor do quantil é obtido através da interpolação linear dos valores dos centróides correspondentes usando seus pesos. Para a consulta da CDF o processo de estimativa do t-digest ocorre de maneira análoga ao da função Quantil, só que fazendo o caminho inverso.

4

Método de aproximação proposto

Neste capítulo, iremos mostrar mais detalhes do método que estamos propondo para realizar a aproximação do cálculo de profundidade de banda utilizando a estrutura de dados t-digest, proposta por DUNNING; ERTL (2019) (Seção 3.5).

A ideia por trás da utilização do t-digest para o cálculo aproximado é usar a estimativa de quantil obtida por essa estrutura de dados para aproximar o *ranking* da curva em determinado ponto, de forma semelhante ao proposto por SUN; GENTON; NYCHKA (2012), só que sem a necessidade de uma matriz para realizar essa aproximação, como veremos mais adiante esse método permite uma boa aproximação, em tempo eficiente e com um custo de memória adicional muito menor.

Para isso criamos um vetor, onde cada um dos seus elementos é um t-digest que tem como responsabilidade obter os centróides para a distribuição de todas as curvas em determinado ponto, com um total de p t-digests, onde p é o número de pontos ds curvas. Na Figura 4.1, por exemplo, teríamos um vetor com cinco t-digests, $T = \{t_1, t_2, t_3, t_4, t_5\}$, um para cada instante de tempo contemplado pelo conjunto de dados.

A construção dos t-digests que compõem o vetor T é feita seguindo o processo mencionado no Capítulo 3, a partir de um vetor de valores que representa os valores das n curvas no ponto j. Uma vez construído o vetor T, utilizamos cada um dos p t-digests para estimar o *ranking* de cada curva no ponto j, onde $0 \leq j \leq p$. Para o cálculo da noção original de profundidade de banda é necessário, então, obter os *rankings* máximo e mínimo das curvas, isso é feito através do Algoritmo 5.

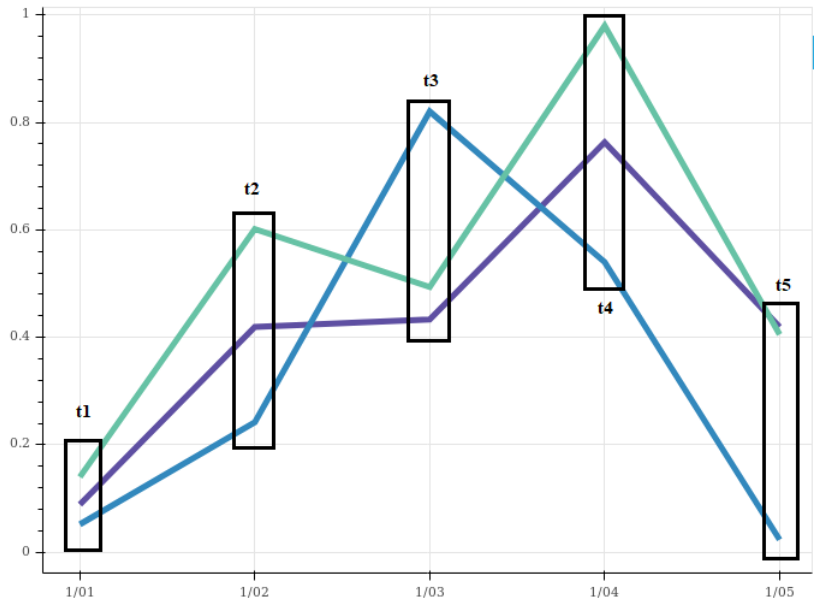


Figura 4.1: Representação dos elementos do vetor de t-digests para um conjunto de curvas. Fonte: do autor.

Algoritmo 5 Algoritmo para achar *rankings* máximo e mínimo das curvas utilizando T

Input: Um conjunto de curvas C com n curvas de tamanho p e T um vetor de t-digests

```
for j = 0 to p − 1 do
    for i = 0 to n − 1 do
        rank ← (T[j].findQuantile(C[i].values[j]) ∗ n) + 1
        if rank > C[i].maxRank then
            C[i].maxRank = rank
        end if
        if rank < C[i].minRank then
            C[i].minRank = rank
        end if
    end for
end for
```

Com os *rankings* obtidos, o cálculo da definição original de profundidade de banda ocorre da mesma forma que no método de SUN; GENTON; NYCHKA (2012), através das Equações (3.8), (3.9) e (3.10).

Da mesma forma, a profundidade de banda modificada também pode ser aproximada utilizando essa abordagem. Em vez de usar a matriz de *rankings*, aproximamos os *rankings* das curvas em cada um dos pontos para achar as proporções das curvas através do Algoritmo 6.

Podemos ver no Algoritmo 6 que cada elemento do *array proportion* representa a proporção de tempo que a curva correspondente está contida nas bandas que não contêm ela na fronteira, da mesma forma que o termo $\frac{\sum_{1\leq i\leq p} n_a(i)n_b(i)}{p}$ na Equação (3.13), e o cálculo da profundidade de banda modificada é feito de forma análoga.

Algoritmo 6 Algoritmo para achar proporções das curvas de um conjunto usando T

Input: Um conjunto de curvas C com n curvas de tamanho p, T um vetor de t-digests e $proportion$, um array de tamanho n

```
match ← matrix_pn
for j = 0 to p − 1 do
    for i = 0 to n − 1 do
        rank ← (T[j].findQuantile(C[i].values[j]) ∗ n) + 1
        n_a ← n − rank
        n_b ← rank − 1
        match[j][i] ← n_a ∗ n_b
    end for
end for
for i = 0 to n − 1 do
    sum ← 0
    for j = 0 to p − 1 do
        sum ← sum + match[j][i]
    end for
    proportion[i] ← sum/p
end for
```

Com o auxílio da estimativa fornecida pelos t-digests nos Algoritmos 5 e 6 conseguimos realizar a aproximação das duas noções de profundidade de banda descritas no Capítulo 3, uma comparação entre os resultados obtidos por essa abordagem, os da abordagem *naïve* e os da abordagem proposta por SUN; GENTON; NYCHKA (2012) será feita no Capítulo 5.

5

Análise experimental

Este capítulo apresenta os experimentos realizados para validar a eficácia da abordagem proposta no Capítulo 4. Para tanto, comparamos o método proposto neste trabalho com as implementações *naïve* e com o método proposto por SUN; GENTON; NYCHKA (2012).

5.1 Metodologia

Nesta seção iremos descrever a metodologia utilizada para os experimentos realizados visando validar o método de aproximação proposto no Capítulo 4.

5.1.1 Os Dados

Para realizar os experimentos, utilizamos quatro conjuntos de dados. Três desses conjuntos são dados a respeito de corridas de táxi na cidade de Nova York: [1] no ano de 2018 (365 dias), nos anos de 2016 a 2018 (3 anos, 1096 dias) e no período de 2014 a 2018 (5 anos, 1826 dias). Esses conjuntos de dados apresentam diversas informações sobre corridas de táxi em Nova York, dentre as informações apresentadas estão a localização do início da corrida, data e hora do início da corrida, número de passageiros, tipo de pagamento, etc. Para este trabalho utilizamos a coluna de data e hora do início da corrida para obter a quantidade de viagens em cada hora de cada dia. O único valor faltante nos três conjuntos de dados das corridas de táxis ocorreu às 2h da manhã do dia nove de março de 2014, que são o dia e a hora em que começa o horário de verão nos Estados Unidos, resultando em uma inconsistência no registro dos dados que ocorreu apenas nesse ano. Esse valor foi substituído pela mediana dos valores do mesmo dia e hora de todos os outros anos.

O outro conjunto de dados, disponibilizado pela plataforma UCI [2], utilizado no trabalho é o resultado da média entre medidas de qualidade de ar captadas por cinco sensores localizados em uma cidade na Itália para todas as horas do dia durante o período de março de 2004 até abril de 2005, totalizando 391 dias. A medida de qualidade de ar que escolhemos para estar

[1] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
[2] https://archive.ics.uci.edu/ml/datasets/Air+quality#

presente no conjunto de dados foi a concentração de monóxido de carbono (CO) em mg/m^3. Nesse conjunto havia um número mais elevado de valores faltantes, que foram substituídos pela mediana dos valores para aquela hora do dia em todos os outros dias. Escolhemos utilizar a mediana em ambos os casos para substituir os valores faltantes por ser uma estatística mais robusta a ruídos e valores extremos.

5.1.2 Comparação dos Métodos de Cálculo de Profundidade de Banda

Para avaliar o quão bem os métodos propostos no Capítulo 3, Seção 3.3 e no Capítulo 4 são capazes de aproximar os resultados dos métodos originais de cálculo de profundidade de banda apresentados no capítulo Capítulo 3, Seção 3.1 e Seção 3.2, nós executamos cada um dos métodos em cada um dos conjuntos de dados detalhados na Seção anterior. A implementação dos algoritmos de cálculo de profundidade de banda foi feita usando a linguagem C/C++. No caso do método proposto por esse trabalho, foi utilizada a implementação de LARA PAHINS; FERREIRA; COMBA (2019) da estrutura de dados t-digest para realizar a aproximação dos *rankings* dos valores das curvas.

5.1.3 Comparação do Custo Computacional das Implementações dos Métodos de Cálculo de Profundidade de Banda

Nesse experimento nosso propósito foi de comparar os dois aspectos computacionais mais importantes das implementações dos algoritmos de cálculo de profundidade de banda: o desempenho de tempo de cada algoritmo e também o custo de memória.

5.1.3.1 Medindo o Desempenho de Tempo dos Algoritmos

Na medição do desempenho de tempo dos algoritmos utilizamos duas abordagens: a primeira foi de medir o tempo levado para calcular a profundidade de banda para cada banda, enquanto a segunda foi medir o tempo de execução da função que calcula a profundidade de banda de todas as curvas de cada conjunto de dados.

Para medir o tempo que a profundidade de banda original de cada curva levou para ser calculada em cada um dos métodos implementados, realizamos a medição de tempo das seguintes formas: para o método original foi medido o tempo de checar, para cada par de curvas, se a curva sendo anaisada estava completamente contida na banda delimitada por cada par; para o método utilizando a matriz de *rankings* foi medido o tempo da operação de achar o *ranking* máximo e mínimo de cada curva e o cálculo de quantas bandas contêm a curva; no método utilizando o t-digest foi medido o tempo de obter os *rankings* máximo e mínimo de cada curva utilizando a estimativa do t-digest, o cálculo de quantas bandas contêm a curva.

Já para medir a performance dos algoritmos no cálculo da profundidade de banda modificada, as seguintes meedições foram feitas: para o método original medimos tempo de calcular

a proporção da curva sendo analisada que estava contida em cada um dos possíveis pares de curvas que delimitavam as bandas do conjunto; no método utilizando a matriz de *rankings* medimos o tempo para obter a proporção de tempo que a curva está contida nas bandas que não contêm ela na fronteira; e por útlimo, no algoritmo usando o t-digest foi medido o tempo de calcular a proporção mencionada anteriormente utilizando as estimatias de *rankings* fornecidas pela estrutura de dados.

Além das medições citadas nos parágrafos anteriores, em todos os casos medimos também o tempo da divisão pelo número total de bandas do conjunto, $\binom{n}{2}$, onde n é o número de curvas existentes no *dataset*. Também medimos o tempo de construção das estruturas de dados adicionais, nos métodos que as utilizam, a matriz de *rankings* e o vetor de t-digests.

5.1.3.2 Medindo o Custo de Memória dos Algoritmos

A medição do custo de memória para cada um dos algoritmos que testamos no trabalho foi feita de acordo com a necessidade de cada um: no caso dos métodos originais de de cálculo de profundidade de banda e profundidade modificada não utilizam memória adicional, de modo que nesses algoritmos não há *overhead* de memória; já no método rápido proposto por SUN; GENTON; NYCHKA (2012) há a utilização de uma matriz de *rankings*, que é uma estrutura cujo tamanho depende dos dados que estão sendo utilizados, no nosso caso, o número de curvas n e o número de pontos por curva p, resultando em um custo de memória de $O(np)$, sendo de nosso interesse observar o tamanho dessa matriz em conjuntos de dados com diferentes números de observações; já a abordagem utilizando t-digests, como visto no Capítulo 4, necessita de um t-digest para cada um dos instantes de tempo contemplados pelo conjunto, sendo de nosso interesse observar como o custo de memória dessa estrutura se comporta ao receber *datasets* de diferentes tamanhos.

5.2 Resultados Experimentais

Iremos nos referir aos conjuntos de dados utilizados nos experimentos através dos seguintes acrônimos: *dataset* de qualidade do ar (AQ), *dataset* de corridas de táxis em 2018 (TX18), *dataset* de corridas de táxis nos anos 2016-2018 (TX1618), *dataset* de corridas de táxis nos anos 2014-2018 (TX1418); os algoritmos de cálculo de profundidade de banda original serão referidos por meio dos seguintes acrônimos: método *naïve* (OD), aproximação utilizando matriz de *rankings* (FD) aproximação utilizando t-digests (TD); e os algoritmos para calcular a profundidade de banda modificada: método *naïve* (OMD), aproximação utilizando matriz de *rankings* (FMD), aproximação utilizando t-digests (TMD). Os experimentos listados neste capítulo foram relizados em uma máquina com sistema operacional Ubuntu 18.04.2 LTS, 16GB de memória RAM e processador Intel Core i7-5500U, CPU @ 2.40 GHz x 4.

5.2.1 Comparação dos Resultados dos Algoritmos

Os gráficos na Figura 5.1 mostram a distribuição dos valores de profundidade de banda para cada um dos métodos analisados nesse trabalho através de *boxplots*. Podemos observar que, para essa definição de profundidade, predominam valores baixos, isso se deve ao fenômeno que foi mencionado no Capítulo 3, de que quando o conjunto possui muitas curvas irregulares o número de bandas que contêm completamente uma curva é pequeno, resultando em um valor pequeno de profundidade e também em muitas curvas com o mesmo valor. É possível ver esse fato através do tamanho do IQR - a "caixa" dos *boxplots* - que é, em geral, pequeno, especialmente para o método *naïve* de cálculo de profundidade de banda. Além disso também é notável que há uma discrepância entre os valores obtidos pelos algoritmos, especialmente entre o método tradicional e os métodos de aproximação estudados, com os últimos tendo um comportamento muito parecido entre si, com uma dispersão maior dos valores em relação ao método tradicional, mostrando que a aproximação realizada pelos métodos possui falhas.

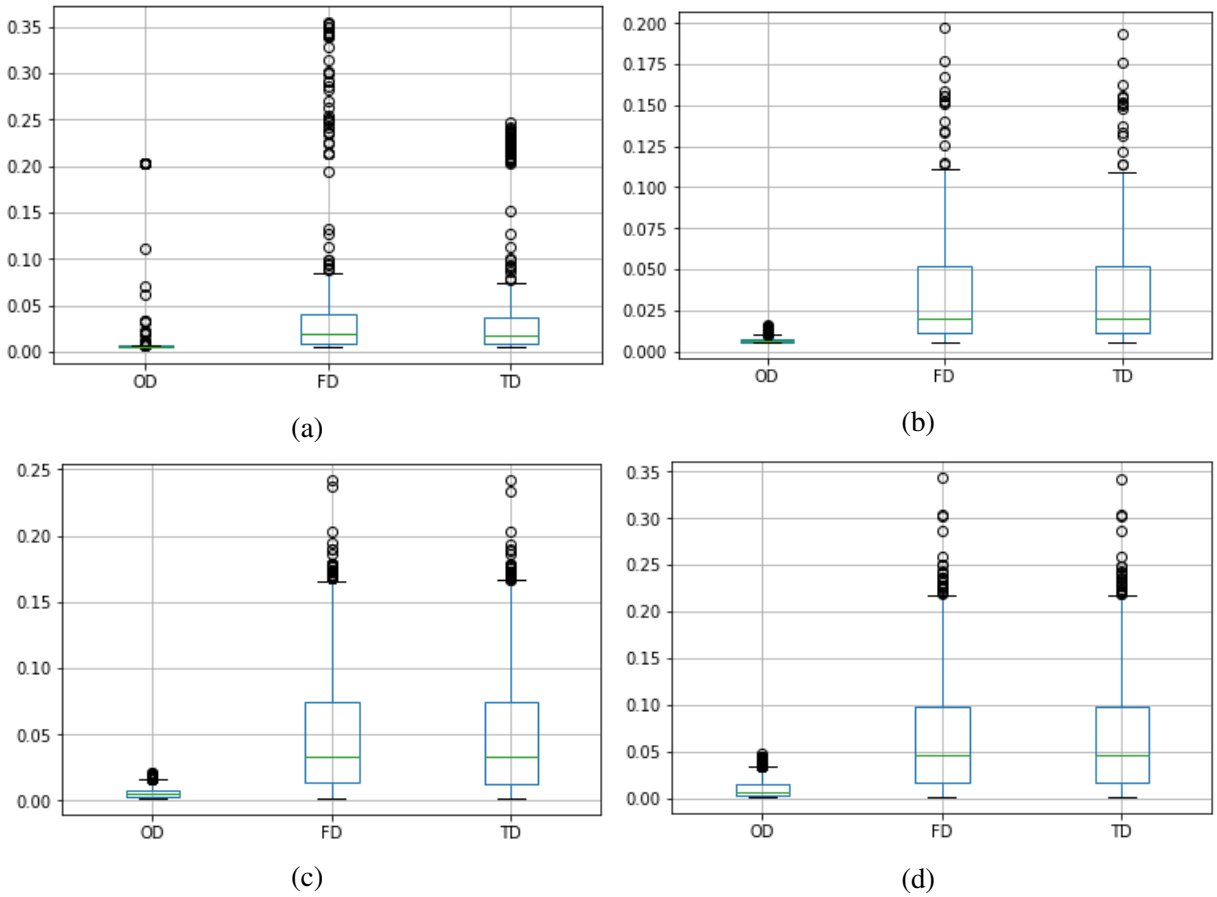


Figura 5.1: *Boxplots* representando a distribuição dos valores de profundidade de banda obtidos por cada um dos algoritmos analisados. Em (a) vemos a distribuição das profundidades de banda originais para AQ, em (b) para TX18, em (c) para TX1618 e em (d) para TX1418 Fonte: do autor.

Já no cálculo da profundidade modificada, podemos observar nos gráficos da Figura 5.2 um comportamento muito mais regular entre todos os algoritmos, o que evidencia uma maior

robustez dessa definição em relação à definição original de profundidade de banda. No gráfico que representa a qualidade do ar, 5.2a, ainda podemos ver uma certa diferença entre o método original e as aproximações, com o método original retornando valores maiores. Entretanto nos gráficos que mostram as distribuições de resultados para os conjuntos de dados referentes a corridas de táxis - 5.2b para um ano, 5.2c para três anos e 5.2d para cinco anos - o comportamento dos três algoritmos foi muito semelhante, demonstrando que a aproximação realizada pelos dois métodos implementados tem um resultado muito próximo do algoritmo original.

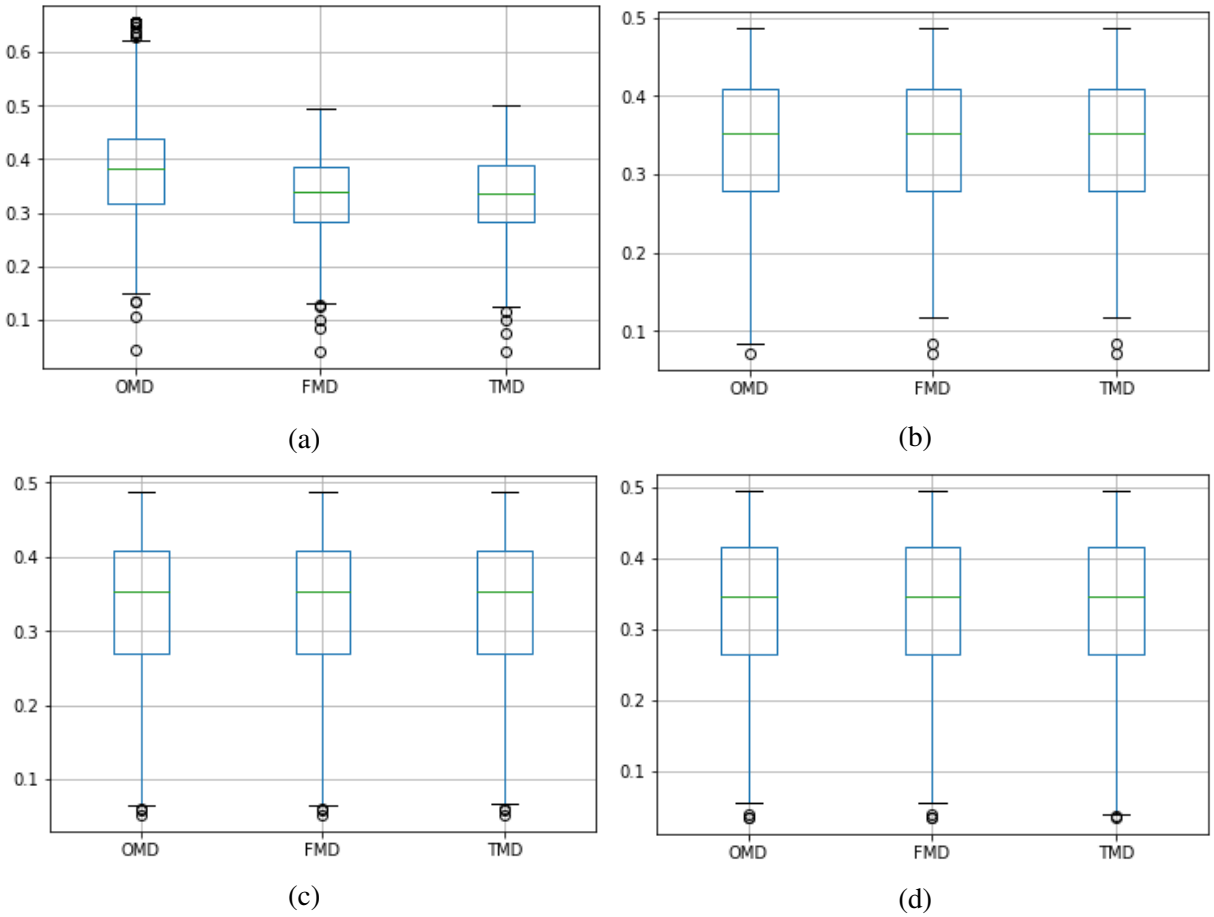


Figura 5.2: *Boxplots* representando a distribuição dos valores de profundidade de banda modificada obtidos por cada um dos algoritmos analisados. Em (a) vemos a distribuição das profundidades de banda modificadas para AQ; em (b) para TX18, em (c) para TX1618 e em (d) para TX1418. Fonte: do autor.

Podemos ver o erro relativo de cada uma das aproximações implementadas em relação às implementações *naïve* para cada um dos conjuntos de dados na Figura 5.3. A coluna da esquerda representa o erro relativo das aproximações da noção original de profundidade de banda em relação à sua implementação *naïve*, enquanto a da direita mostra a mesma coisa, só que pra noção de profundidade de banda modificada.

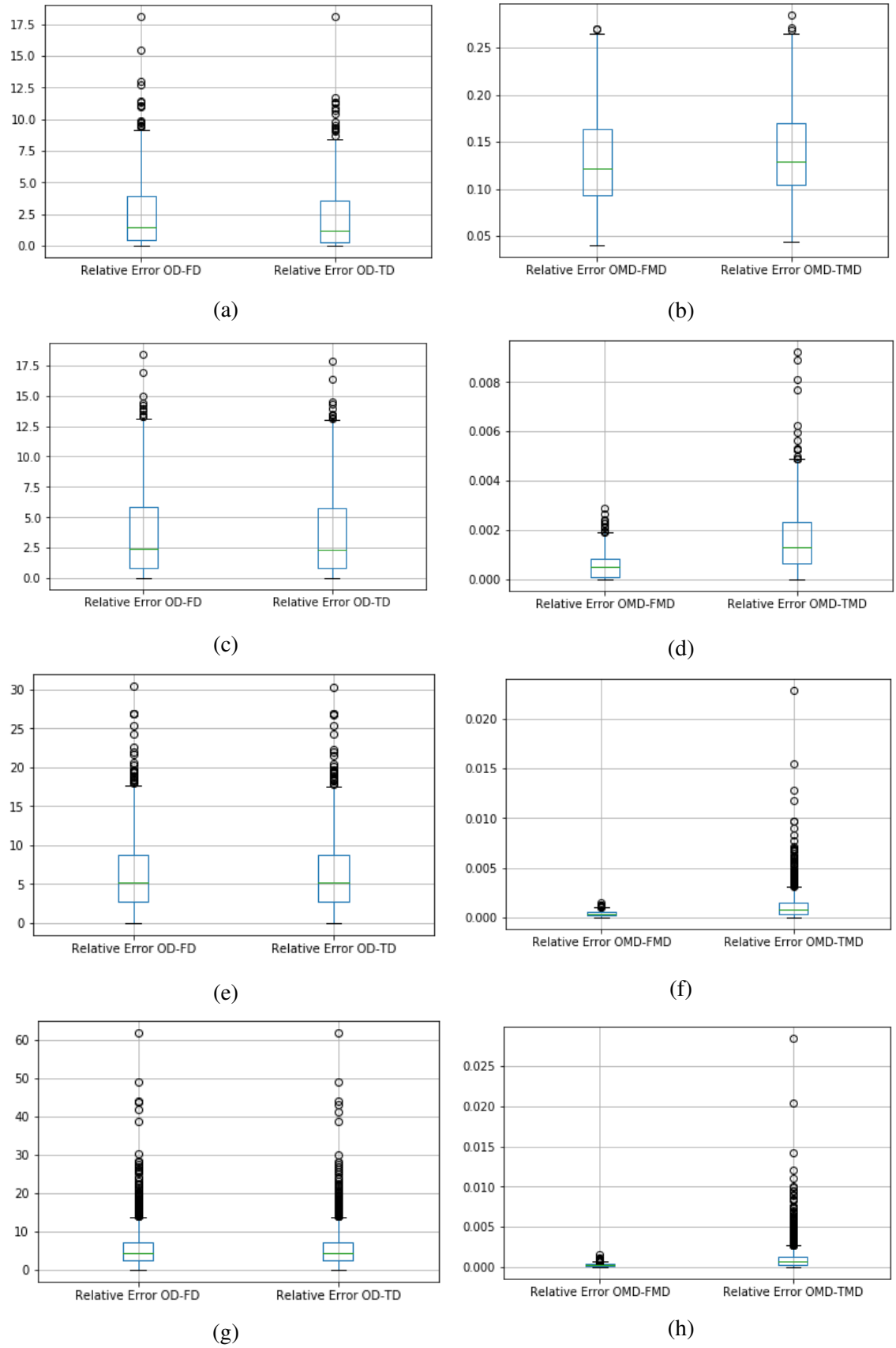


Figura 5.3: Distribuição do erro relativo dos métodos de aproximação em relação à implementação *naïve* para ambas as noções de profundidade de banda. (a) Erro relativo da BD para AQ. (b) Erro relativo da MBD para AQ. (c) Erro relativo da BD para TX18. (d) Erro relativo da MBD para TX18. (e) Erro relativo da BD para TX1618. (f) Erro relativo da MBD para TX1618. (g) Erro relativo da BD para TX1418. (g) Erro relativo da MBD para TX1418. Fonte: do autor.

É possível observar que, as distribuições dos erros relativos da noção original de profundidade de banda (representada pela coluna da esquerda na Figura 5.3) têm valores muito mais altos do que as distribuições da noção de profundidade modificada (coluna da direita na Figura 5.3), reforçando a ideia de que a noção original é mais sensível do que a modificada. Por sua vez, as distribuições dos erros relativos para a profundidade modificada possuem valores muito baixos, mostrando que as aproximações para essa definição tiveram um desempenho muito próximo da implementação *naïve* exata. Vemos que, apesar de o erro relativo do método proposto por esse trabalho, TMD, ser maior do que a aproximação de SUN; GENTON; NYCHKA (2012), ele se mantém sempre muito baixo, com a grande maioria não chegando nem a 1% nos datasets das corridas de táxis, TX18, TX1618 e TX1418.

Outra forma de visualizar a diferença entre a implementação *naïve* da profundidade de banda e as aproximações utilizando a matriz de *rankings* e utilizando os t-digests é através dos *functional boxplots* dos resultados obtidos por cada uma dessas abordagens. Vemos nas Figuras 5.4a, 5.4c e 5.4e que há uma diferença perceptível entre o número de *outliers* detectados no conjunto dos dias de semana em TX18 pelo método naïve - OD - e as aproximações FD e TD, que classificaram muito mais curvas como *outliers*, representadas pelas curvas em laranja claro e também pelas curvas que aparecem na legenda em vermelho, azul, verde, laranja e roxo, as cinco com menor profundidade obtidas por cada método.

Podmos ver, também através dos *functional boxplots* que as aproximações da noção de profundidade de banda modificada têm um resultado muito mais condizente com a implementação exata dessa noção do que as aproximações da profundidade de banda original e sua implementação exata. Essa relação é mostrada nas figuras na coluna da direita na Figura 5.4 nas quais podemos observar que os gráficos obtidos a partir dos resultados de cada método detectaram os mesmos outliers, demonstrando a maior robustez desse método em comparação a noção original de profundidade de banda, que é muito sensível a ruídos nos dados, como foi explicado no Capítulo 3. Mais detalhes a respeito dos *functional boxplots* serão apresentados no Capítulo 6.

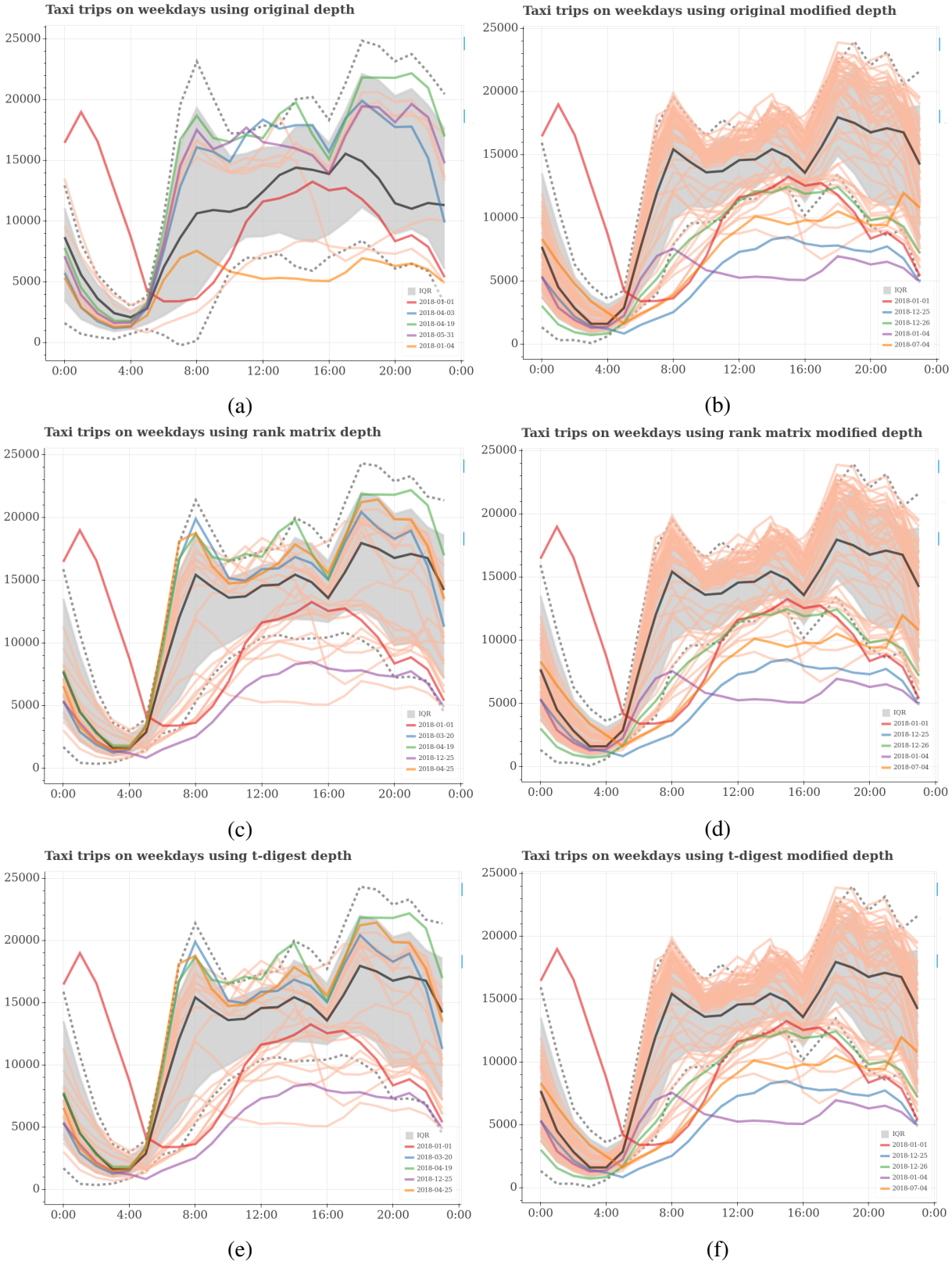


Figura 5.4: *Functional boxplots* dos resultados dos cálculos da profundidade de banda e profundidade de banda modificada obtidos por cada método nos dias de semana em TX18. (a) *Functional boxplot* dos resultados de OD. (b) *Functional boxplot* dos resultados de OMD. (c) *Functional boxplot* dos resultados de FD. (d) *Functional boxplot* dos resultados de FMD. (e) *Functional boxplot* dos resultados de TD. (f) *Functional boxplot* dos resultados de TMD. Fonte: do autor.

5.2.2 Comparação do Desempenho dos Algoritmos

Nesta seção fazemos a comparação em termos de tempo de execução dos algoritmos para o cálculo da profundidade de banda.

A Tabela 5.1 mostra o tempo (em milissegundos) - que cada um dos algoritmos levou para calcular a profundidade de banda original, BD. Podemos perceber que a performance do método *naïve*, como dito no Capítulo 3, piora significativamente com o aumento no número de curvas, fato evidenciado pelo grande aumento do tempo nos *datasets* referentes às corridas de táxis: TX18 tem 365 curvas, TX1618 tem 1096 curvas e TX1418 tem 1826 curvas, com o tempo deste último chegando próximo de dois minutos. O desempenho do método de aproximação rápido utilizando a matriz de *rankings* tem o melhor desempenho entre os algoritmos analisados dado que, uma vez construída a matriz, o tempo para calcular a profundidade de uma curva depende apenas de achar os *rankings* máximo e mínimo, que é linear sobre o número de pontos da curva. Entre os dois métodos mencionados anteriormente está o algoritmo proposto por esse trabalho que, embora não seja tão rápido quanto o método utilizando a matriz, tem um desempenho muito melhor do que o algoritmo *naïve*. A Tabela 5.2 reforça essa ideia, mostrando o tempo médio que cada algoritmo leva para calcular a profundidade de banda de uma curva nos conjuntos analisados.

Dataset	OD	FD	TD
AQ	1145,82ms	0,68ms	14,19ms
TX18	700,81ms	0,64ms	13,37ms
TX1618	21155,35ms	2,48ms	51,90ms
TX1418	115815,50ms	4,06ms	83,23ms

Tabela 5.1: Tempo levado por cada um dos algoritmos para calcular as profundidades de todas as curvas. Fonte: do autor.

Dataset	OD	FD	TD
AQ	2,92ms	< 0,001ms	0,02ms
TX18	1,91ms	0,001ms	0,02ms
TX1618	19,30ms	0,001ms	0,03ms
TX1418	63,42ms	0,001ms	0,03ms

Tabela 5.2: Tempo médio para calcular a profundidade de banda de uma curva em cada um dos algoritmos. Fonte: do autor.

Da mesma maneira que foi feito para o cálculo da profundidade original, também medimos o tempo para o cálculo da profundidade modificada. As Tabelas 5.3 e 5.4 mostram o desempenho de cada um dos algoritmos nos *datasets* estudados. Assim como nos algoritmos de cálculo da noção original de profundidade de banda, o método *naïve* teve a pior performance entre os três - chegando a demorar cerca 12 minutos e meio em TX1418 -, o método proposto utilizando os t-digests teve uma performance comparável ao método utilizando a matriz de *rankings*, que foi o mais rápido.

Dataset	OMD	FMD	TMD
AQ	7711,30ms	1,01ms	14,52ms
TX18	5589,16ms	0,94ms	17,39ms
TX1618	174569,38ms	3,66ms	51,01ms
TX1418	750286,62ms	6,02ms	85,27ms

Tabela 5.3: Tempo levado por cada um dos algoritmos para calcular as profundidades modificadas de todas as curvas. Fonte: do autor.

Dataset	OD	FD	TD
AQ	19,72ms	< 0,001ms	0,02ms
TX18	15,31ms	0,001ms	0,03ms
TX1618	159,27ms	0,001ms	0,03ms
TX1418	410,88ms	0,001ms	0,03ms

Tabela 5.4: Tempo médio para calcular a profundidade de banda modificada de uma curva em cada um dos algoritmos. Fonte: do autor.

Podemos notar através das tabelas que, para o método *naïve*, o tempo para cálculo da profundidade modificada é muito superior do que o da profundidade original o que mostra que a obtenção da proporção das curvas que estão contidas nas bandas é um processo que demora mais do que a verificação se a curva está completamente contida, nesse método. O algoritmo rápido proposto por SUN; GENTON; NYCHKA (2012) também mostra um aumento significativo para o cálculo da profundidade modificada, chegando a aumentar em cerca de 50% em relação ao cálculo da original. Enquanto isso vemos que o método utilizando os t-digests é o que apresenta as menores diferenças entre os tempos para calcular a profundidade modificada e a original, exemplificando a versatilidade dessa abordagem.

Outro ponto importante de ser observado é o tempo levado para a construção de cada uma dessas estruturas de dados adicionais. A Tabela 5.5 mostra, em milissegundos, o tempo para a construção das estruturas de dados necessárias para o cálculo da profundidade para os dois métodos que necessitam, é importante reiterar que a performance da construção da matriz de *rankings* é dependente do algoritmo de ordenação utilizado para obter os *rankings*, por conveniência optamos por utilizar o algoritmo *bubble sort*, resultando em uma complexidade de tempo total da ordem de $O(n^2 p)$.

Dataset	FD / FMD	TD / TMD
AQ	20,02ms	3,69ms
TX18	25,83ms	3,75ms
TX1618	292,58ms	15,19ms
TX1418	830,99ms	26,15ms

Tabela 5.5: Tempo para construção de estruturas de dados necessárias para o cálculo dos métodos de aproximação de profundidade de banda. Fonte: do autor.

5.2.3 Comparação do Custo de Memória dos Algoritmos

Além da medição do tempo, também medimos o custo de memória adicional para cada um dos algoritmos testados. Com isso, nosso objetivo foi de analisar o *trade-off* entre o desempenho do algoritmo e quanta memória adicional ele precisa para fazer o cálculo. A Tabela 5.6 mostra, em *kilobytes*, o custo de memória de cada um dos algoritmos analisados nessa etapa.

Dataset	FD / FMD	TD / TMD
AQ	37,54 *kB*	19,48 *kB*
TX18	35,04 *kB*	19,48 *kB*
TX1618	105,21 *kB*	19,48 *kB*
TX1418	175,29 *kB*	19,48 *kB*

Tabela 5.6: Custo de memória adicional - para os métodos que a utilizam - em cada um dos conjuntos de dados analisados. Fonte: do autor.

Observando os valores na tabela é notória a eficiência do método proposto utilizando os t-digests em relação ao algoritmo rápido e sua matriz de *rankings*. A matriz é uma estrutura que vai crescer com o número de dados, tendo sempre np elementos, onde n é o número de curvas e p o número de pontos por curva. Enquanto isso o método proposto por este trabalho que utiliza p vetores de t-digests para realizar a aproximação dos *rankings* não sofre com o aumento de curvas, se mantendo em um valor constante independente do crescimento dos conjuntos de dados analisados.

Podemos ver que, além de levar um tempo maior para ser construída, a matriz de *rankings* proposta por SUN; GENTON; NYCHKA (2012) também ocupa mais espaço, em relação ao algoritmo utilizando as estruturas t-digest proposto por nós.

5.3 Discussão

Analisando tanto a complexidade de tempo quanto o custo de memória adicional, podemos fazer um balanço dos três algoritmos analisados nessa seção: apesar de não precisar de nenhum tipo de memória adicional, o método *naïve* de LÓPEZ-PINTADO; ROMO (2009) para o cálculo de profundidade de banda se mostra extremamente lento a medida que o número de amostras aumenta, tornando sua utilização em grandes coleções de dados impraticável; o método proposto por SUN; GENTON; NYCHKA (2012) se mostrou o que tem a melhor performance, entretanto o seu custo de memória adicional é muito grande, o que resulta em um gasto de memória exagerado para conjuntos que possuam muitas curvas ou muitos pontos; por último o método proposto por esse trabalho utilizando a estrutura de dados t-digest apresentada no trabalho de DUNNING; ERTL (2019), que teve um desempenho comparável ao método utilizando a matriz de *rankings* e com a vantagem que não sofreu alterações drásticas com o crescimento da quantidade de dados e além disso se mostrou ser muito mais eficiente em termos de memória, sendo o mais equilibrado entre os três. Outra vantagem do método proposto

por este trabalho acontece em cenários de uso que exijam filtragens e agregações de conjuntos de dados diferentes, a estrutura proposta é a mais adequada dado que t-digests diferentes podem ser combinados.

6

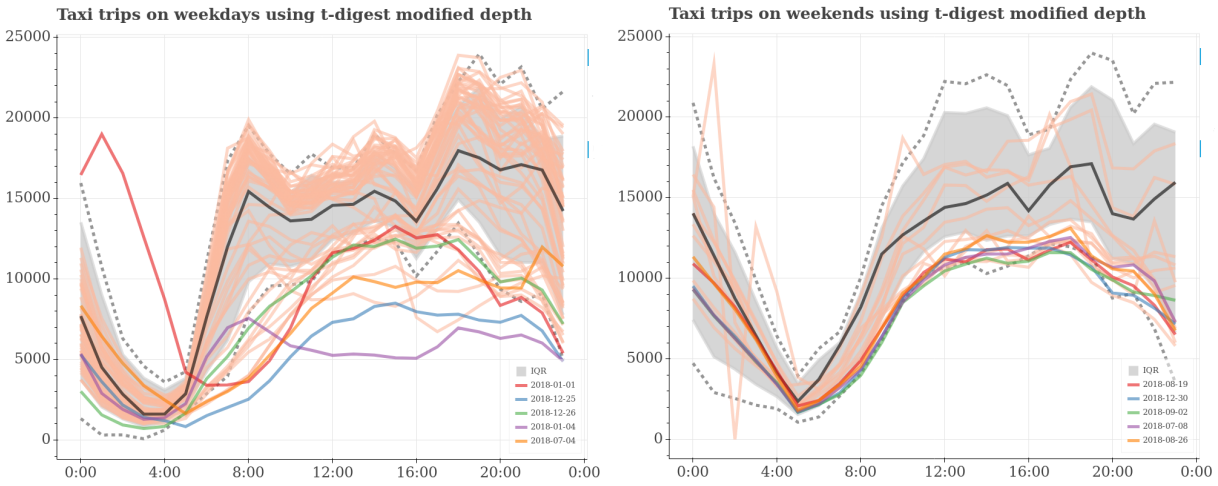
Casos de Uso

Neste capítulo, mostraremos como o conceito de profundidade de banda pode ser utilizado em uma técncia de visualização de dados. Para isso iremos mostrar *functional boxplots*, apresentados por SUN; GENTON (2011), utilizando os valores de profundidade de banda modificada obtidos pelo algoritmo proposto no Capítulo 4 e que é uma aproximação, como foi visto do Capítulo 5. Escolhemos utilizar a profundidade modificada nesse caso por ser uma definição mais robusta a ruídos, como foi explicado anteriormente no Capítulo 3 e comprovado no Capítulo 5. Dividimos o conjunto TX18 em dois subconjuntos - um representando os dias de semana e outro representando os finais de semana - com o objetivo de observar a distribuição das curvas em cada um desses subconjuntos e utilizar as características do *functional boxplot*, bem como as noções de profundidade de banda para observar curvas com o comportamento diferente do padrão e que podem ser consideradas *outliers*. A implementação do *functional boxplot* utilizado neste capítulo foi feita utilizando a biblioteca Bokeh [1] de Python.

Vemos na Figura 6.1 os *functional boxplots* mencionados anteriormente. Neles a banda cinza representa o IQR, ou seja, os 50% centrais dos dados em cada ponto do gráfico; enquanto isso, as linhas cinza tracejadas acima e abaixo do IQR denotam o envelope máximo sem *outliers*, que, como o nome diz, marca o limite do comportamento aceitável de uma curva; a curva preta é a curva mediana, a que tem a maior profundidade de banda modificada; e as curvas em laranja claro denotam todas as curvas que, em algum momento, ultrapassam os limites do envelope máximo sem *outliers*, sendo consideradas possíveis *outliers*, dentre elas existem cinco curvas com cores diferentes (vermelho, azul, verde, roxo e laranja) que são as cinco que têm as menores profundidades de banda modificada dentre todas as 365 do conjunto, cujas datas de cada uma podem ser vistas na legenda.

Analisando a distribuição dos dados no gráfico dos dias de semana na Figura 6.1a podemos ver que há um declínio grande no número de viagens de táxi durante a madrugada, bem como um declínio na variabilidade do número de viagens ilustrado pelo encolhimento do IQR, seguido de um aumento brusco entre as 5h e 8h. Entre 8h e 10h há um novo declínio e aí a distribuição não demonstra uma variação significativa até as 16h, em que há uma nova queda

[1] https://docs.bokeh.org/en/latest/



(a) (b)

Figura 6.1: *Functional boxplots* dos resultados do cálculo da profundidade de banda modificada obtidos pelo algoritmo proposto nos dois subconjuntos de TX18. Em (a) estão os resultados do método TMD para os dias de semana e em (b) para os finais de semana Fonte: do autor.

brusca, isso se deve ao fato de que as 16h ocorre uma troca de turno entre os taxistas, onde eles voltam para a garagem e novos pilotos assumem a frota. Após a troca de turno há um novo aumento no número de viagens entre as 16h e 19h, horário em que tipicamente as pessoas estão voltando do trabalho. Nesse gráfico podemos ver, dentre os cinco maiores *outliers*, a curva vermelha - representando dia primeiro de Janeiro - na contramão de todos os outros dias com um aumento no número de viagens de táxi na madrugada decorrente da movimentação do dia de ano novo. Além disso também vemos os dias 25 e 26 de Dezembro, as curvas azul e verde, respectivamente, com um número mais baixo de viagens de táxi do que o normal, provavelmente devido às festas natalinas. O dia quatro de Julho, dia da independência dos Estados Unidos também se encontra entre as cinco curvas com menor profundidade representado pela curva laranja. Além dessas datas comemorativas, o dia 4 de Janeiro de 2018 também está entre esses *outliers* com um número muito mais baixo de viagens de táxi ao longo do dia devido a uma tempestade de neve que atingiu a costa leste americana nesse dia. Enquanto isso a curva com maior profundidade, ou seja, a curva mediana, ocorreu no dia 9 de Agosto.

Já observando a distribuição das corridas de táxi nos finais de semana, observamos na Figura 6.1b que há um número maior de viagens no início da madrugada, evidenciado pelo posicionamento mais alto do IQR e um declínio um pouco menos abrupto do que o do *functional boxplot* dos dias de semana. Assim como o declínio, o aumento no início da manhã também é mais suave nos finais de semana. Da mesma forma que nos dias de semana, por volta das 16h há uma diminuição abrupta nas corridas, decorrente da mesma troca de turno dos taxistas. Podemos observar que nesse conjunto há duas curvas com comportamentos significativamente diferente de todas as outras, uma com um pico muito alto durante a madrugada e outra com um vale muito acentuado. Nesse caso, como esses valores ocorrem apenas em um ponto das curvas enquanto os demais se encontram dentro da normalidade, a profundidade de banda modificada

não é tão afetada por valores extremos como a definição original. As cinco curvas com menor profundidade de banda se encontram muito mais próximas do limite inferior do envelope máximo sem *outliers* nesse conjunto do que no conjunto contemplando os dias de semana. Nesse conjunto a curva mediana é representada pelo dia 6 de Outubro.

7

Conclusão

A visualização de grandes coleções de séries temporais é um tópico ativo na academia. Técnicas clássicas como o gráfico de linhas e gráfico de linhas empilhadas (ambos propostos por PLAYFAIR (1801)) sofrem com o problema de *overplotting* e a técnica de *small multiples* (proposta por TUFTE; GOELER; BENSON (1990)) mostra-se impraticável para um número alto de séries temporais. Por isso, novas alternativas para a visualização de conjuntos com um grande número de séries temporais são propostas, como *DenseLines* de MORITZ; FISHER (2018), que se utiliza do conceito de densidade como ferramenta de visualização, permitindo ver áreas com maior ou menor concentração de linhas. No entanto, essa técnica não é capaz de identificar comportamentos anômalos em regiões de maior densidade ou padrões em regiões de densidade menor, sendo necessária uma nova abordagem na visualização de gráficos de linha com muitas séries temporais.

Vimos que o conceito de PDF é muito útil na análise de dados funcionais, pois ele permite a ordenação desses dados baseado na centralidade em uma distribuição. Das noções de PDF, uma das mais clássicas é a profundidade de banda LÓPEZ-PINTADO; ROMO (2009), cujos métodos tradicionais de computação não são adequados para utilização em sistemas de visualização interativa de dados por não terem um bom desempenho para conjuntos grandes de dados. Outra maneira de realizar esse cálculo foi apresentada por SUN; GENTON; NYCHKA (2012), porém ela utiliza muita memória. É importante que o tempo de execução e o custo de memória em sistemas de visualização de dados não sejam muito elevados, como visto em LIU; HEER (2014).

Desse modo, apresentamos uma nova abordagem para o cálculo aproximado das noções de profundidade de banda, a qual consiste na utilização de uma estrutura de dados chamada t-digest, apresentada no tabalho de DUNNING; ERTL (2019). Essa estrutura é capaz de computar aproximações de estatísticas de ordem com alta precisão e com um *overhead* de memória adicional muito menor que o método proposto por SUN; GENTON; NYCHKA (2012).

Podemos ver, através dos experimentos, que o método proposto neste trabalho fornece uma aproximação precisa do conceito de profundidade de banda modificada. Além disso, nosso método obteve um desempenho muito melhor do que os algoritmos *naïve* e, apesar de não ser

tão rápido quanto o método proposto por SUN; GENTON; NYCHKA (2012), tem um custo de memória bem reduzido, mesmo com o aumento dos conjuntos de dados que foram testados. Ademais, vimos que não há diferença significativa na performance de tempo do cálculo da profundidade original em relação ao cálculo da profundidade modificada, fato que ocorre nos outros métodos testados, evidenciando a versatilidade do algoritmo proposto.

Observamos a utilização do conceito de profundidade de banda modificada em uma técnica de visualização de dados através de casos de uso envolvendo um conjunto de dados referente a corridas de táxi na cidade de Nova York. Através disso mostramos como esse conceito pode ser útil na identificação e visualização de curvas com comportamento distinto, mesmo com um grande número de curvas na coleção.

Em futuros trabalhos, ainda há a possibilidade de diminuir mais o custo de memória adicional do t-digest, através do parâmetro de compressão δ, que controla o número máximo de centroides. Além disso, também é possível realizar a aproximação das noções de profundidade de banda por meio de outras abordagens. Um desses metódos pode ser através da utilização de técnicas de aprendizagem de máquina ou *deep learning*, que já foi usada para fazer outros tipos de agregação em técnicas de cubos de dados no trabalho de WANG et al. (2018).

Referências

AIGNER, W. et al. Time & time-oriented data. In: **Visualization of Time-Oriented Data**. [S.l.]: Springer, 2011. p.45–68.

DORAISWAMY, H. et al. Using topological analysis to support event-guided exploration in urban data. **IEEE transactions on visualization and computer graphics**, [S.l.], v.20, n.12, p.2634–2643, 2014.

DUNNING, T.; ERTL, O. Computing extremely accurate quantiles using t-digests. **arXiv preprint arXiv:1902.04023**, [S.l.], 2019.

FULCHER, B. D.; LITTLE, M. A.; JONES, N. S. Highly comparative time-series analysis: the empirical structure of time series and their methods. **Journal of the Royal Society Interface**, [S.l.], v.10, n.83, p.20130048, 2013.

GRAY, J. et al. Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub-totals. **Data mining and knowledge discovery**, [S.l.], v.1, n.1, p.29–53, 1997.

JAVED, W.; MCDONNEL, B.; ELMQVIST, N. Graphical perception of multiple time series. **IEEE transactions on visualization and computer graphics**, [S.l.], v.16, n.6, p.927–934, 2010.

LARA PAHINS, C. A. de; FERREIRA, N.; COMBA, J. Real-Time Exploration of Large Spatiotemporal Datasets based on Order Statistics. **(Pre-print) IEEE Transactions on Visualization and Computer Graphics**, [S.l.], p.1–1, 2019.

LIU, Z.; HEER, J. The effects of interactive latency on exploratory visual analysis. **IEEE transactions on visualization and computer graphics**, [S.l.], v.20, n.12, p.2122–2131, 2014.

LÓPEZ-PINTADO, S.; ROMO, J. On the concept of depth for functional data. **Journal of the American Statistical Association**, [S.l.], v.104, n.486, p.718–734, 2009.

MIRANDA, F. et al. Time lattice: a data structure for the interactive visual analysis of large time series. In: COMPUTER GRAPHICS FORUM. **Anais...** [S.l.: s.n.], 2018. v.37, n.3, p.23–35.

MORITZ, D.; FISHER, D. Visualizing a Million Time Series with the Density Line Chart. **arXiv preprint arXiv:1808.06019**, [S.l.], 2018.

PHILLIPS, J. M. Coresets and sketches. **arXiv preprint arXiv:1601.00617**, [S.l.], 2016.

PLAYFAIR, W. **The commercial and political atlas**: representing, by means of stained copper-plate charts, the progress of the commerce, revenues, expenditure and debts of england during the whole of the eighteenth century. [S.l.]: T. Burton, 1801.

SUN, Y.; GENTON, M. G. Functional boxplots. **Journal of Computational and Graphical Statistics**, [S.l.], v.20, n.2, p.316–334, 2011.

SUN, Y.; GENTON, M. G.; NYCHKA, D. W. Exact fast computation of band depth for large functional datasets: how quickly can one million curves be ranked? **Stat**, [S.l.], v.1, n.1, p.68–74, 2012.

TUFTE, E. R.; GOELER, N. H.; BENSON, R. **Envisioning information**. [S.l.]: Graphics press Cheshire, CT, 1990. v.126.

WANG, Z. et al. NNCubes: learned structures for visual data exploration. **arXiv preprint arXiv:1808.08983**, [S.l.], 2018.