UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
INSTITUTO MULTIDISCIPLINAR

RAUL SENA FERREIRA

# WSAGE - Ferramenta Web para Análise de Dados Geoespaciais

Prof. Carlos Eduardo R. de Mello, Ph.D.
Orientador

Nova Iguaçu, Dezembro de 2014

# WSAGE - Ferramenta Web para Análise de Dados Geoespaciais

**Raul Sena Ferreira**

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto Multidisciplinar da Universidade Federal Rural do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação. Apresentado por:

---

Raul Sena Ferreira

Aprovado por:

---

Prof. Carlos Eduardo R. de Mello, Ph.D.

---

Prof. Fellipe Ribeiro Duarte , M.Sc.

---

Prof. Filipe Braida do Carmo, M.Sc.

NOVA IGUAÇU, RJ - BRASIL

Dezembro de 2014

# Agradecimentos

Quero agradecer, em primeiro lugar, a Deus, por iluminar o meu caminho e me dar força durante toda a caminhada.

Dedico esta conquista, aos meus amados "pais-avós"(José de Moura e Maria do Céu), aos meus queridos pais (Kim e Meire), meu irmão (Bruno), minhas irmãs (Gabrielle, Camilly e Maria Júlia) e a todos os meus primos que para mim são como irmãos.

Dedico também a minha esposa Rafaella, pelo apoio, dedicação e persistência nestes 12 anos de relacionamento, compreendendo minhas faltas e acreditando em mim desde o início dessa jornada.

Agradeço também a todos os professores que me acompanharam durante a graduação, em especial ao Prof. Dr. Carlos Eduardo R. Mello, que sempre fez o possível para atender as demandas dos alunos e do curso além de ser o responsável pela realização deste trabalho, ao Prof. M.Sc. Filipe Braida do Carmo por ter aberto as portas para UFRJ, lugar onde trabalho e futuramente continuarei meus estudos, ao Prof. M.Sc. Fellipe Duarte pela ótima pessoa que é e pela gentileza de compor esta banca.

Por último, e não menos importante, dedico este trabalho à todos os meus colegas e amigos(as) que fiz durante a graduação, se eu fosse mencionar todos não caberia nesta página, sem eles eu também não teria conseguido chegar até aqui, obrigado à todos vocês!!!

RESUMO

WSAGE - Ferramenta Web para Análise de Dados Geoespaciais

Raul Sena Ferreira

Dezembro/2014

Orientador: Carlos Eduardo R. de Mello, Ph.D.

Estimar densidades sempre foi importante nos mais diversos cenários, é uma técnica estatística que ajuda a determinar, de forma mais precisa, a probabilidade sobre determinados tipos de dados.

Por sua vez, existem dados que não são puramente discretos mas que podem revelar muito sobre uma população, no caso, os dados geográficos. Este trabalho busca criar uma interface online intuitiva para análise desses dados em conjunto com dados não espaciais, utilizando um método estimador de densidade e assim, disponibilizar ao usuário da ferramenta, percepções importantes sobre a base de dados observada.

ABSTRACT

WSAGE - Ferramenta Web para Análise de Dados Geoespaciais

Raul Sena Ferreira

Dezembro/2014

Advisor: Carlos Eduardo R. de Mello, Ph.D.

*Estimate density has always been important in various scenarios, is a statistical technique that helps determine, more precisely, the probability of certain types of data.*

*In its turn, there are data that are not purely discrete but can reveal much about a population, the geographic data. This work seeks to create an intuitive online interface for data analysis in conjunction with non-spatial data, using an estimator method of density and thus, make available to the user, important perceptions about the database observed.*

# Lista de Figuras

# Lista de Tabelas

# Lista de Códigos

# Lista de Abreviaturas e Siglas

| | |
|---|---|
| UFRRJ | Universidade Federal Rural do Rio de Janeiro |
| W-SAGE | Web tool for Space Analysis of Geodata |
| JSON | JavaScript Object Notation |
| AJAX | Asynchronous Javascript and XML |
| HTML | HyperText Markup Language |
| CSS | Cascading Style Sheets |
| DOM | Document Object Model |
| PHP | Hypertext Preprocessor |
| XML | eXtensible Markup Language |
| SQL | Structured Query Language |
| KDE | Kernel Density Estimation |
| GPU | Graphics Processing Unit |

# Lista de Símbolos

| | |
|---|---|
| UFRRJ | Universidade Federal Rural do Rio de Janeiro |
| W-SAGE | Web tool for Space Analysis of Geodata |
| JSON | JavaScript Object Notation |
| AJAX | Asynchronous Javascript and XML |
| HTML | HyperText Markup Language |
| CSS | Cascading Style Sheets |
| DOM | Document Object Model |
| PHP | Hypertext Preprocessor |
| XML | eXtensible Markup Language |
| SQL | Structured Query Language |

# Sumário

# Capítulo 1

# Introdução

## 1.1 Motivação

A informação geográfica tem grande importância em diversas áreas como, marketing, agricultura, meio ambiente, saúde, planejamento urbano entre outros, ajudando na tomada de decisões e estratégias bem como agregando valor como um meio de representação visual mais expressiva do que uma representação discreta.
Várias ferramentas podem ser criadas visando extrair o máximo de informações agregadas a distribuição espacial, informações essas que não poderiam ser extraídas através do modo convencional de análise de dados não espacial. A informação geográfica, já existe há centenas de anos, (e.g., mapas) e como em vários aspectos do nosso cotidiano esta também foi e vem sendo alterada pela modernidade tecnológica. Para lidar com esse tipo de informação, surgiram então os sistemas de informação geográfica (SIG). Os SIG são sistemas utilizados para armazenar, analisar, manter e manipular dados geográficos de maneira automatizada [3]. Os dados geográficos utilizados pelos SIG podem ser imagens digitalizadas (e.g., fotos de satélite) ou objetos que representam uma geometria no espaço, chamados objetos espaciais.
Esses dados são armazenados e gerenciados por bancos de dados espaciais (objetos geométricos espaciais). [5] E é aí que essas ferramentas se encaixam bem com o objetivo deste trabalho, que visa criar uma ferramenta para análise de dados geoespaciais usando diversas técnicas e tecnologias combinadas, entre elas um banco

de dados geográfico, permitindo experimentar um tipo de visualização que não só mostre os dados geográficos mas também tenha algum método estatístico como base para a visualização.

## 1.2 Objetivos Gerais

O objetivo deste trabalho é desenvolver uma Ferramenta Web de análise espacial de dados geográficos, para principalmente estimar densidades sobre um dado não-paramétrico qualquer, baseado em uma prévia consulta espacial com ou sem restrição de região (polígono).
Esta ferramenta também deveria permitir a visualização dessa densidade estimada em formato de mapa de calor (Heatmap) e deveria permitir que se visualizasse os dados de forma geolocalizada (pontos). Alguns gráficos também seriam propostos para enriquecer ainda mais a análise.
A ferramenta foi desenvolvida para que profissionais do setor referente ao dado analisado ou pessoas interessadas pela informação extraída da ferramenta possam analisar os dados e usar a ferramenta de forma intuitiva.

## 1.3 Objetivos Específicos

O objetivo em específico é poder usar a ferramenta no banco de dados de alunos matriculados da UFRRJ e partir daí estimar densidades e extrair informação de valor, que com o auxílio das componentes geográficas, podem trazer percepções sobre o rendimento escolar, evasão ou até mesmo saber se a localização da faculdade de fato atende a região para qual ela foi intencionalmente criada, tudo isso baseado na região em que o aluno reside e nas informações escolares desses alunos. Outro objetivo considerado importante é que a ferramenta processasse e mostrasse os dados com velocidade satisfatória em uma interface clara e organizada.

## 1.4 Estrutura

Este trabalho está organizado em 6 capítulos, onde este é o primeiro. No segundo capítulo é apresentada a proposta escolhida para ser utilizada na ferramenta. O terceiro capítulo possui um detalhamento dos conceitos e tecnologias envolvidas no desenvolvimento do trabalho. Já no quarto capítulo, detalhamos os desafios e os passos tomados para a construção da ferramenta juntamente com o método estimador proposto. Atavés do quinto capítulo mostramos os resultados obtidos e fazemos algumas comparações com outra ferramenta já consolidada. E por último, o sexto capítulo, consiste em mostrar as conclusões, considerações finais e trabalhos futuros.

# Capítulo 2

# Estimando Densidades

Há vários métodos na literatura de estimação de densidade de probabilidade, dentre eles o método mais conhecido é o estimador de densidade de kernel (ou Kernel Density Estimation), também conhecido como Janela de Parzen[1]. Neste método são utilizadas funções não-lineares como Gaussianas e Sigmóides para se computar a densidade local de cada instância.
Estimar densidades sobre uma população é um trabalho importante e geralmente usamos uma função de densidade para isso. A densidade de uma população pode ser estimada com várias técnicas estatísticas, porém estatisticamente, alguns dados ou populações não possuem estruturas ou parâmetros característicos, no caso, estes dados são conhecidos como não paramétricos.
A função de probabilidade é um conceito fundamental em estatística e existem diversas técnicas que podem ser empregadas para estimar dados não paramétricos e a técnica escolhida para este trabalho foi o KDE(Kernel Density Estimation), devido a simplicidade e a já conhecida eficiência perante a literatura para tratar esses tipos de dados.

## 2.1 Dados paramétricos x dados não paramétricos

Os dados podem ser paramétricos ou não paramétricos. Os dados paramétricos são provenientes de um tipo de distribuição de probabilidade e faz inferências sobre

os parâmetros da distribuição. A maioria dos métodos elementares estatísticos são paramétricos.

Métodos paramétricos fazem mais suposições que os métodos da estatística não paramétrica. Se essas suposições extras são corretas, métodos paramétricos podem produzir estimativas mais precisas. Elas possuem maior potência estatística.

Porém, se as hipóteses estão incorretas, métodos paramétricos podem se tornar falhos. Por esse motivo, eles são considerados menos robustos. Um exemplo de dado paramétrico: a distribuição normal tem uma média e uma variância especificados.

Já os dados não paramétricos não dependem de dados pertencentes a nenhuma distribuição particular. Tipicamente, o modelo não-paramétrico cresce no sentido de acomodar a complexidade dos dados. Como métodos não paramétricos fazem menos suposições, a aplicabilidade deles é mais larga que os correspondentes métodos paramétricos. Em particular, eles podem ser aplicados em situações em que menos se sabe sobre o problema em questão. Além disso, devido a menor dependência de hipóteses, métodos não paramétricos são mais robustos. Um exemplo de dado não-paramétrico: distribuição tem a forma normal, tanto a média quanto a variância não foram especificadas.

## 2.2 Kernel Density Estimation

O KDE é uma das técnicas de estimativa de densidade mais comuns e é bastante usada para normalizar e suavizar a distribuição de um determinado conjunto de dados. O KDE pode pode ser pensado como uma generalização do histograma.

Possui duas variações: Univariante, cuja a entrada são dados de uma única dimensão, no caso um vetor; e Multivariante, cuja a natureza dos dados de entrada é de duas ou mais dimensões, usa uma matriz para armazenamento dos dados.

O processamento do algoritmo KDE Multivariável é feito levando-se em consideração cada indivíduo em relação a sua população, e a sua complexidade é $O(n^2k)$, pois é implementado como um somatório de um produtório de matrizes, por tanto, dependendo do tamanho da entrada, o algoritmo pode-se tornar um tanto quanto

lento para apresentar a estimativa final de cada indivíduo.

A saída deste algoritmo é a estimativa de densidade final, mais conhecido como PDF (Probability Density Function), que é calculado para cada indivíduo em relação a sua população.

A partir de um dado número de observações n, calculamos curvas de densidade delas em relação a distância de um valor central, o núcleo, para cada um desses pontos e obtemos a Estimativa de Densidade final somando esses valores.

Um kernel é uma função de ponderação padronizada, ou seja, o núcleo determina a suavização do PDF. A função Kernel $K_h(x, x_t)$ pode ser expressa como um produto interno no espaço de features na forma $K_h(x_a, x_b) = \phi(x_a)^T \phi(x_b)$ onde a função $\phi$ define o mapeamento $\phi : X \to F$. Esta técnica é amplamente usada em vários algoritmos de aprendizado de máquina, principalmente em SVM (Support Vector Machines).

Esta função Kernel precisa ser cuidadosamente escolhida pois pode provocar um super ajustamento ou o contrário nos valores dos PDF [1][2], no caso deste trabalho será utilizado o Kernel Gaussiano, pois produz uma estimativa mais suave, porém há outros tipos de funções, como descrito na figura 2.1.

| *Kernel* | $K(t)$ | *Efficiency (exact and to 4 d.p.)* |
|---|---|---|
| Epanechnikov | $\frac{3}{4}(1-\frac{1}{5}t^2)/\sqrt{5}$ for $\lvert t\rvert < \sqrt{5}$; $0$ otherwise | 1 |
| Biweight | $\frac{15}{16}(1-t^2)^2$ for $\lvert t\rvert < 1$; $0$ otherwise | $\left(\frac{3087}{3125}\right)^{1/2} \approx 0.9939$ |
| Triangular | $1 - \lvert t\rvert$ for $\lvert t\rvert < 1$, 0 otherwise | $\left(\frac{243}{250}\right)^{1/2} \approx 0.9859$ |
| Gaussian | $\frac{1}{\sqrt{2\pi}}\, e^{-(1/2)t^2}$ | $\left(\frac{36\pi}{125}\right)^{1/2} \approx 0.9512$ |
| Rectangular | $\frac{1}{2}$ for $\lvert t\rvert < 1$, 0 otherwise | $\left(\frac{108}{125}\right)^{1/2} \approx 0.9295$ |

Figura 2.1: Tipos de Kernel

Uma implementação do KDE Univariante poderia ser escrita da seguinte forma:

**Algorithm 1** KDE Univariante

**for** $i \leftarrow 0$ *to* $n$ **do**
    soma_kernel $\leftarrow 0.0$
    **for** $j \leftarrow 0$ *to* $n$ **do**
        soma_kernel $\leftarrow K((x[i] - x[j])/h)$
    **end**
    $pdf[i] \leftarrow$ soma_kernel / n
**end**

Já o multivariante:

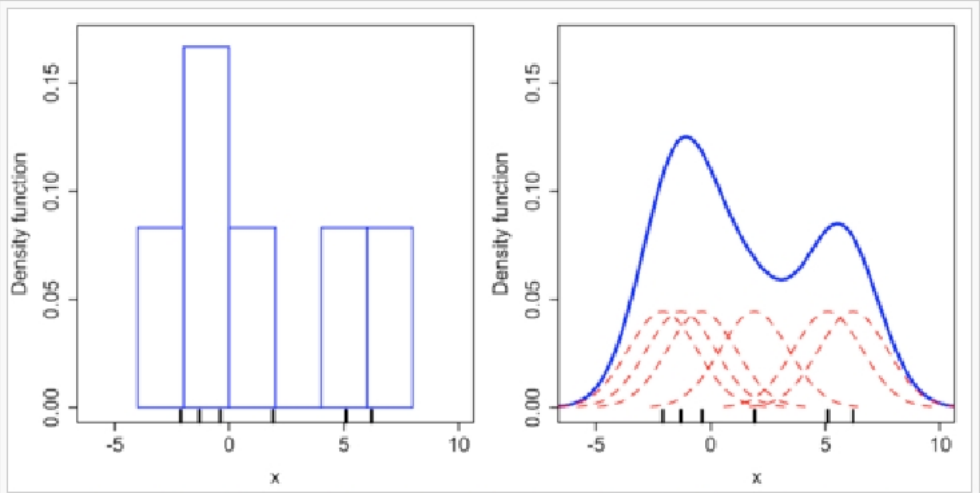
**Algorithm 2** KDE Multivariante

**for** $i \leftarrow 0$ *to* $n$ **do**
    soma_kernel $\leftarrow 0.0$
    **for** $j \leftarrow 0$ *to* $n$ **do**
        prod_kernel $\leftarrow 1.0$
        **for** $k \leftarrow 0$ *to* $xLen$ **do**
            prod_kernel $*K((x[i][k] - x[j][k])/h)/h$
        **end**
        soma_kernel $\leftarrow$ soma_kernel + prod_kernel
    **end**
    $pdf[i] \leftarrow$ soma_kernel / n
**end**

Conforme descrito na introdução o KDE é bastante comum e utilizado em vários problemas, dentre eles:

- Estimativa de Densidade Geográfica de uma região
- Correção de ruídos em sinais elétricos
- Análise estática de dados
- Estimativa de densidade populacional

A figura 2.2 mostra um exemplo de diferença entre um histograma convencional e um suavizado pelo uso do KDE:




Comparison of the histogram (left) and kernel density estimate (right) constructed using the same data. The 6 individual kernels are the red dashed curves, the kernel density estimate the blue curves. The data points are the rug plot on the horizontal axis.

Figura 2.2: histograma x kde

### 2.2.1 KDE Python

Para esta ferramenta fizemos o uso do KDE Gaussiano implementado pela biblioteca SciPy desenvolvida especialmente para cálculos científicos em Python. Em seguida, invocamos para cada ponto da distribuição o método evaluate do objeto kernel, dessa forma obtemos o valor do PDF.

Os parâmetros, com exceção do vetor N-Dimensional contendo a população a ser estimada, são opcionais, e são descritos a seguir:

dataset: os dados que serão estimados.

bw_method : parâmetro usado para suavizar o kernel gaussiano, pode ser determinado pelo método de Scott, Silverman ou um escalar, no caso deste trabalho foi escolhido o método padrão, ou seja, o método de Scott. O código completo do KDE do SciPy pode ser encontrado em seu repositório público em

https://github.com/scipy/scipy/blob/master/scipy/stats/kde.py

# Capítulo 3

# Conceitos básicos e tecnologias envolvidas

## 3.1 Introdução

O projeto foi desenvolvido usando dados coletados das matrículas de todos os alunos que ingressaram na UFRRJ desde 2000 até o fim de 2013. Os dados são guardados no SGBD, no caso deste trabalho, o Postgres, que em conjunto com o plugin PostGIS, fica responsável pelo armazenamento e pelo processamento de consultas espaciais realizadas pela ferramenta.

O processamento no lado servidor é feito com a linguagem PHP e Python, rodando em um servidor web (Apache), onde o PHP fica responsável por interligar uma requisição feita pelo usuário com o banco de dados, além de chamar o método estimador escrito em Python, que por sua vez, será o responsável pelo processamento do algoritmo e enviará o resultado para o cliente.

No lado cliente, onde é feita a visualização dos dados, a ferramenta utiliza, embutida em páginas .php, a linguagem HTML5 em conjunto com a linguagem JavaScript. A estilização dos dados é feita com CSS3.

## 3.2 Tecnologias envolvidas

O sistema precisa ser interativo, moderno e eficaz como ferramenta e para isso a escolha certa das ferramentas e tecnologias fazem grande diferença no tempo de desenvolvimento, manutenção, desempenho, visual e na escalabilidade da aplicação.

### 3.2.1 Bancos de dados espaciais

Nesta seção será apresentada uma visão geral dos bancos de dados espaciais e o sistema gerenciador de banco de dados escolhido para este trabalho.
Os bancos de dados geográficos são coleções de dados georreferenciados, manipulados por um sistema de informação geográfica (SIG). Os SIG necessitam que o banco de dados dê suporte à consultas que utilizem operações e propriedades espaciais, além do suporte para os dados relacionais(não-espaciais).
Os bancos de dados geográficos utilizados pelo SIG possuem dados usualmente agrupados em duas componentes: a componente espacial(geográfica) e a componente convencional.[6] A componente espacial de um SIG geralmente dividi-se em:
Ponto; Linha; Polígono;
O tipo Ponto é utilizado para representar um objeto no espaço, indicando sua localização. O tipo Linha geralmente é utilizado para representar rotas no espaço, como rodovias, rios, rotas de linhas de ônibus, trajetos origem - destino, entre outros. O tipo Polígono é utilizado quando queremos representar formas e áreas no espaço, como, cidades, países, regiões etc. Os bancos de dados espaciais fornecem suporte a consultas utilizando operações sobre relações espaciais entre os objetos. Essas relações podem ser de três tipos [5].
Relações topológicas como toca, adjacente, dentro, disjunto e sobrepõe. Estas são independentes de transformações nos objetos como translação, escala ou rotação. Relações de direção como acima, abaixo, norte de, sul de, etc. Relações de medida como "distância $>$ 100" ou "área $>$ 100". O banco de dados espacial utilizado neste trabalho foi o PostGIS em conjunto com o SGBD Postgres.

#### *3.2.1.1 PostGIS*

PostGIS é uma base de dados espacial extendida para uso com o banco de dados relacional PostgreSQL. O PostGIS suporta objetos geográficos permitindo que consultas relacionadas a geolocalização sejam realizadas usando SQL.
PostGIS também tem uma série de vantagens como:
Funções de processamento e de análise para o vetor e dados raster para emendar, cortar, reclassificar, álgebra de mapas para o processamento raster, reprojeção espacial, suporte para importação/exportação de dados vetoriais shapefile ESRI via linha de comando e/ou GUI, suporte para mais formatos através de outras ferramentas Open Source.
Importa dados raster a partir de vários formatos padrão: GeoTiff , NetCDF , PNG , JPG e etc. Renderização e importação de dados vetoriais para formatos textuais padrão, como KML , GML , GeoJSON , Geohash e WKT. Suporte a objeto 3D, índice espacial e funções de suporte a topologia de rede.

### 3.2.2 Redis

Redis (REmote DIctionary Server) é um banco de dados muito popular, open source, que guarda os registros em formato chave-valor todos em memória com durabilidade opcional.
É ótimo para ser usado como cache de aplicações. Possui diversas estruturas de dados como, hash, listas e métodos para recuperar, guardar, deletar, atualizar e percorrer dados, além de permitir clusterização, replicação entre outras soluções.

### 3.2.3 Servidor HTTP Apache

É um servidor web open source, responsável por aceitar requisições HTTP dos navegadores e servi-los com resposta HTTP, além de permitir a execução de programas e scripts pré-configurados.
É neste servidor que os arquivos da aplicação executam as funções acessando a base de dados eventualmente trazendo o resultado para o browser.

Suporta arquivos tanto estáticos, como o HTML, quanto dinâmicos, no caso PHP e Python. Funciona em sistemas operacionais UNIX e Windows. É o servidor web mais popular do mundo.

### 3.2.4 PHP

É uma linguagem interpretada livre, usada originalmente apenas para o desenvolvimento de aplicações presentes e atuantes no lado do servidor, capazes de gerar conteúdo dinâmico na web. O código é interpretado no lado do servidor pelo módulo PHP, que também gera a página web a ser visualizada no lado do cliente.

### 3.2.5 Python

Python é uma linguagem de programação de alto nível, interpretada, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. Foi desenvolvido para ser uma linguagem de fácil leitura, com um visual agradável, frequentemente usando palavras e não pontuações como em outras linguagens.
Para a separação de blocos de código, a linguagem usa espaços em branco e indentação ao invés de delimitadores visuais como chaves (C, Java) ou palavras (BASIC, Fortran, Pascal). Diferente de linguagens com delimitadores visuais de blocos, em Python a indentação é obrigatória. O aumento da indentação indica o início de um novo bloco, que termina da diminuição da indentação. Esta linguagem também é open source e possui diversos módulos e bibliotecas efecientes, principalmente para cálculos científicos.

#### *3.2.5.1 SciPy*

É uma biblioteca em python com diversas funções para lidar com cálculos científicos. Possui código-fonte livre e uma numerosa comunidade de desenvolvedores e usuários. Ela traz consigo outra biblioteca muito útil em cálculos matemáticos, o NumPy.

#### *3.2.5.2 NumPy*

É uma biblioteca que facilita o uso e o cálculo envolvendo vetores N-Dimensionais.

#### *3.2.5.3 Joblib*

É uma biblioteca que disponibiliza uma série de ferramentas para facilitar a paralelização de aplicações em Python.

### 3.2.6 HTML5

O HTML é uma linguagem de marcação web amplamente difundido, é utilizado para produzir páginas na web. Arquivos feitos com essa linguagem são interpretados pelos navegadores. Todo documento HTML possui tags, palavras entre parênteses angulares(< e >). As tags básicas do HTML são: <html>: define o início de um documento HTML e indica ao navegador que todo conteúdo posterior deve ser tratado como uma série de códigos HTML <head>: define o cabeçalho de um documento HTML, que traz informações sobre o documento que está sendo aberto <body>: define o conteúdo principal, o corpo do documento. Esta é a parte do documento HTML que é exibida no navegador. No corpo podem-se definir atributos comuns a toda a página, como cor de fundo, margens, e outras formatações. O HTML está atualmente em sua quinta versão, com novos recursos, antes só possíveis em conjunto com outras tecnologias. Sua essência tem sido melhorar a linguagem com o suporte para as mais recentes multimídias, enquanto a mantém facilmente legível por seres humanos e consistentemente compreendida por computadores e outros dispositivos (móveis, parsers, etc). O HTML5 é apontado como o novo padrão para HTML, XHTML, e HTML DOM. Atualmente, está em fase de esboço, porém diversos navegadores já implementam várias de suas funcionalidades. Este projeto usa algumas das novas tags do HTML5, dentre elas a mais importante na confecção do mapa, o elemento <canvas> </canvas>, que torna possível a manipulação de elementos gráficos e de multimídia sem o uso de APIs proprietárias.

### 3.2.7 CSS3

O CSS3 facilita o trabalho de desenvolvedores e usuários pela variedade de transformações na apresentação de um site. É extremamente capaz de construir animações tanto em 2 como em 3 dimensões.

### 3.2.8 JavaScript

JavaScript é uma linguagem de programação interpretada. Foi originalmente implementada como parte dos navegadores web para que scripts pudessem ser executados do lado do cliente e interagissem com o usuário sem a necessidade deste script passar pelo servidor, controlando o navegador, realizando comunicação assíncrona e alterando o conteúdo do documento exibido.
É atualmente a principal linguagem para programação client-side em navegadores web. Começa também a ser bastante utilizada do lado do servidor através de ambientes como o node.js. Foi concebida para ser uma linguagem script com orientação a objetos baseada em protótipos, tipagem fraca e dinâmica e funções de primeira classe. Possui suporte à programação funcional e apresenta recursos como fechamentos e funções de alta ordem comumente indisponíveis em linguagens populares como Java e C++.

### 3.2.9 Bibliotecas utilizadas

#### *3.2.9.1 JQuery*

jQuery é uma biblioteca JavaScript cross-browser e de código aberto desenvolvida para simplificar os scripts client side que interagem com o HTML.
A sintaxe do jQuery foi desenvolvida para tornar mais simples a navegação do documento HTML, a seleção de elementos DOM, criar animações, manipular eventos e desenvolver aplicações AJAX. A biblioteca também oferece a possibilidade de criação de plugins sobre ela. As principais funcionalidades incluem, resolução da incompatibilidade entre os navegadores, redução de código, reutilização do código

através de plugins, utilização de uma vasta quantidade de plugins criados por outros desenvolvedores, trabalha com AJAX e DOM, implementação segura de recursos do CSS1, CSS2 e CSS3.

#### *3.2.9.2 JQueryUI*

jQuery UI é uma biblioteca para criar interações de interface, efeitos, widgets e temas de forma rápida, usando como base a biblioteca jQuery.

#### *3.2.9.3 Bootstrap*

Bootstrap é um framework para desenvolvimento web front-end (HTML, CSS, e JavaScript), é responsivo (o visual se adequa bem tanto para desktops quanto para dispositivos móveis).
Bootstrap ajuda a desenvolver de forma mais rápida aplicativos e websites.

#### *3.2.9.4 OpenLayers*

O OpenLayers (OL) é uma biblioteca open source JavaScript que pode ser usada para disponibilizar/exibir dados geográficos na internet.
O OL pode obter dados de diversos recursos, tais como os padrões do OGC Web Map Service (WMS), Web Feature Service (WFS) bem como Google Maps, OpenStreetMap, Bing Maps, MapServer, GeoServer e muitos outros. Possui ainda suporte à GeoRSS, navegação via mouse e pelo teclado, adição de marcadores e seleção de layers.

#### *3.2.9.5 Chart.js*

Chart.js é uma biblioteca JavaScript que usa dados digitais para criar e controlar gráficos interativos em páginas web. É uma ferramenta de visualização de dados fazendo uso de Vetores Gráficos Escaláveis (SVG), JavaScript, HTML5, e CSS3.

#### *3.2.9.6 HeatMap*

É uma biblioteca como o próprio nome diz, feita para criar mapas de calor em volta de um ponto no espaço. Ele funciona da seguinte forma:
Por padrão, para cada ponto no mapa ele atribui uma cor correpondente ao peso um, ou seja, ele atribui uma cor azul claro de raio x. As cores se sobrepõe ficando mais fortes (azul, passando para amarelo e chegando até o vermelho) caso haja interseção entre os raios, ou seja, se houver muitos pontos em uma região próxima, a tendência é que aquela região atinja uma coloração vermelha (quente) e as regiões de menor densidade de pontos ficará azulado (frio).
O funcionamento da biblioteca é intuitivo, de fácil integração e possui certa facilidade de customização para quem conhece um pouco de javascript.
Ela possui integração com a biblioteca OpenLayers e com o google maps, além de ter um bom desempenho e suportar até 40 mil pontos.

#### *3.2.9.7 Google Maps v3*

A API Javascript do Google Maps permite incorporar o Google Maps em páginas da Web. A Versão 3 desta API foi especialmente desenvolvida para ser mais rápida e mais aplicável aos dispositivos móveis, bem como aos aplicativos de navegadores desktop tradicionais.
A API oferece diversos utilitários para manipulação de mapas (assim como na página da Web http://maps.google.com) e para a adição de conteúdo ao mapa por meio de diversos serviços, o que permite criar robustos aplicativos de mapas.
A API JavaScript do Google Maps v3 é um serviço gratuito, disponível para qualquer website que seja gratuito para os consumidores.

### 3.2.10 Matlab

O MATLAB é um sistema interativo cujo elemento básico de informação é uma matriz que não requer dimensionamento. Esse sistema permite a resolução de muitos

problemas numéricos em apenas uma fração do tempo que se gastaria para escrever um programa semelhante em linguagem Fortran, Basic ou C. Além disso, as soluções dos problemas são expressas quase exatamente como elas são escritas matematicamente.

No caso deste trabalho, o Matlab foi usado como parâmetrizador dos nossos testes, devido a sua alta confiabilidade e robustez.

# Capítulo 4

# Desenvolvimento do Estudo de Caso

## 4.1 Construção da aplicação

A aplicação foi construída em 2 partes: Lado servidor ou mais comumente chamado de Back-End, responsável pelo processamento do KDE, pelas requisições do usuário, além de fazer a ligação entre o banco e a aplicação;
Lado cliente ou simplesmente Front-End, responsável por intermediar a interface do usuário com o servidor além de garantir a visualização das consultas. Abaixo é explicado o desenvolvimento de cada uma dessas 2 partes.

### 4.1.1 Construção do Front-End

O front-end é composto por uma página inicial, onde é dada uma introdução sobre o que é o projeto, as funcionalidades existentes, link para teste, link para o repositório do projeto e disponibilização de um canal de contato com o desenvolvedor. O link de teste leva o usuário para uma página de login para que ele entre com suas credenciais e acesse a aplicação.
Depois de autenticado, o usuário acessa a aplicação através do seu painel de controle, esta possui em grande parte de sua estrutura tags HTML5, estilizadas pelas bibliotecas Bootstrap e jQuery UI, escritas em CSS3 e Javascript, além disso, foi inserido um mapa da API do Google Maps em conjunto com o OpenLayers.js e o

Heatmap.js, onde os dados são plotados em forma de pontos e mapa de calor.
Graças a biblioteca Open Layers, a consulta na aplicação pode ser feita por polígono, delimitando uma região, ou sem polígono, todo o globo terrestre, passando para o back-end o tipo de objeto espacial que está sendo procurado. As requisições das consultas são enviadas por AJAX fazendo o uso da biblioteca jQuery.js além de funções auxiliares escritas em javascript puro.
Depois de enviado a requisição de consulta para o servidor, este recupera os dados e em seguida processa o KDE que por sua vez entrega o resultado, no caso, um vetor de pdf, para a biblioteca heatmap que é inicializada para criar o mapa de calor atribuindo os valores dos PDFs como pesos dos pontos, gerando assim colorações que variam do azul (menor peso) ao vermelho (maior peso) e inserindo através do OpenLayers essa informação no mapa. Enquanto isso, quase que paralelamente, os pontos retornados da consulta são colocados dentro do mapa, gerando assim a visualização dos pontos e do mapa de calor. Logo depois, a biblioteca de gráficos Chart.js pega as informações da consulta em formato JSON e carrega os gráficos correspondentes.
A vantagem de usar javascript é que os pontos são colocados em objetos e assim os pontos que vão ao mapa não são meras coordenadas e sim objetos interativos podendo ter reação a cliques, mouseover entre outros.

### 4.1.2 Clusterização e Rasterização dos pontos

Devido a grande quantidade de pontos no navegador foi preciso pensar em uma estratégia para diminuir o carregamento da página. A saída foi implementar a clusterização dos pontos por distância, ou seja, pontos que estão muito próximos entre si ou que possuem a mesma coordenada não precisam aparecer. Caso o usuário dê o zoom na página então essa distância relativa tende a aumentar e consequentemente os pontos clusterizados devem aparecer, e assim sucessivamente.
Dessa forma evitamos que vários pontos fiquem agrupados em um espaço pequeno ou até mesmo um em cima do outro, dificultando a visualização e deixando a página desnecessariamente carregada. Outro ponto importante na implementação foi o uso

de rasterização dos pontos, utilizando o elemento canvas do HTML5.
Rasterizar uma imagem nada mais é do que converter uma imagem vetorial em uma imagem raster (pixels ou pontos) e colocá-la em buffer. Dessa forma os pontos deixam de ser objetos e caso uma fique sobre a outra a imagem acima tende a sobrescrever o que estiver abaixo dela, evitando que múltiplas imagens empilhadas sejam criados na tela. De posse dessas duas implementações consegue-se plotar mais de 14 mil pontos sem haver quase travamento algum do navegador.

### 4.1.3 Construção do Back-End

O back-end é composto pelo arquivo source.php que é responsável por fazer a consulta no PostGres/PostGIS e também faz a chamada para o script feito em python que executa o KDE com os mesmos parâmetros da consulta enviada para o arquivo .php. A resposta então é enviada para o arquivo source.php, que por sua vez, repassa para o front-end em formato JSON.
As informações contidas nos inputs existentes na view são recebidas no arquivo .php através do método POST, em seguida, a query é construída dinamicamente de acordo com as informações existentes, essas informações nada mais são do que as escolhas feitas pelo usuário no formulário da página de consulta.
A execução do KDE é feito da seguinte forma. As coordenadas são enviadas pro python através do php, em seguida o script faz uma busca no banco e carrega as coordenadas em formato JSON, colocando as informações resultantes em um vetor de objetos, e partir daí, é criado uma matriz de coordenadas, latitude e longitude, que depois é enviado para um método da biblioteca científica scipy, que executa o algoritmo do KDE gaussiano. O retorno é um array de PDF, que logo depois é passado para o PHP e de lá o mesmo envia para o browser. A biblioteca Scipy(Scientific Library for Python) é responsável por processar o KDE gaussiano.

## 4.2 Construção dos Dados de Entrada

Nesta seção é descrito o que foi necessário para se obter os dados de entrada, mostrando todo o processo de tratamento para tornar possível a utilização na aplicação e assim gerar resultados mais precisos.

### 4.2.1 Coleta

Os dados foram coletados junto à diretoria da universidade, em planilha excel, com os dados de matrícula de todos os alunos que se matricularam na universidade de 2000 até 2013. Os dados fornecidos foram cep, situacao da matrícula, status de bolsista, sexo, nascimento, naturalidade, forma de ingresso, periodo real, periodo cronologico, campus, código do curso, cr acumulado e percentual integralizado.
Não houve coleta de nomes ou qualquer tipo de dado sócio-econômico, apesar dos dados serem usados em uma aplicação aberta tomou-se o cuidado em manter o anonimato dos dados dos alunos incluídos no sistema.

### 4.2.2 Conversão e Limpeza

Como dito na subseção anterior, os dados vieram em uma planilha excel e como essas informações foram preenchidas de forma manual acabaram por vir com algumas falhas como, cep incompleto, cep inexistente, matrícula não preenchida, cep não preenchido entre outros. Foi preciso então fazer uma prévia limpeza e correção dos dados.
Primeiramente, foram retirados os registros com cep ou matrícula não preenchidos, depois os ceps incompletos com um dígito a menos foram corrigidos na medida do possível, acrescentando-se um zero no final, tomando-se o cuidado para não alterar a região do aluno.
Em seguida, foi utilizado o site batchgeo.com, para extrair as coordenadas geográficas dos ceps restantes, e foi aí que foram descobertos ceps incorretos, que foram assumidos assim por não serem identificados pelo site, no final da conversão esses

registros também foram retirados.

### 4.2.3 Carga de Dados

Depois de juntar os registros com suas respectivas coordenadas em um arquivo csv, criamos uma tabela no Postgres com os campos id, cep, situacao, bolsista, sexo, nascimento, naturalidade, forma_ingresso, periodo_real, periodo_cronologico, campus, cod_curso, cra, perc_integralizado, geom (geometria), latitude, longitude.
Com o banco pronto, os dados foram importados por psql, enviando as informações contidas no csv para o banco e posteriormente foi executado um update criando os objetos geométricos a partir das coordenadas e em seguida foram criados índices para otimizar a consulta.

## 4.3 Redis - Otimizando o carregamento da aplicação

A aplicação, apesar dos esforços de otimização no carregamento, ainda possuía um gargalo de processamento, no caso, o cálculo do KDE, pois este estava levando em média 14 segundos para carregar pouco mais de 12.500 pontos, e toda a vez que uma nova consulta era feita, um novo processamento era realizado.
Para tentar resolver esse gargalo se fez necessário pensar em alguma estratégia de cache desses PDFs resultantes desses processamentos. Então fizemos o uso do Redis.
No arquivo kde.py, tranformamos o select da consulta em um numero hexadecimal, mais precisamente um md5.
Depois usamos esse md5 como chave de um HashSet contido no Redis, e verificamos se existe algum valor associado a essa chave, se existir, significa que essa consulta já foi feita antes e o resultado já está guardado no banco local do Redis, então temos o trabalho apenas de retornar o resultado sem a necessidade de um reprocessamento do KDE.
Caso o HashSet não retorne nada com a chave dada, então conectamos ao Postgres, fazemos o processamento do KDE com SciPy e em seguida guardamos a chave e

os PDFs associados a essa chave no Redis, em seguida entregamos ao navegador o resultado. Dessa forma, conseguimos fazer com que a aplicação tivesse um ganho de velocidade de até 14 vezes, no caso do processamento com 14.000 pontos.

# Capítulo 5

# Resultados

Como parâmetro foi usado o Matlab para rodar o KDE gaussiano com as mesmas coordenadas usadas na aplicação e depois foi comparado o tempo do algoritmo paralelizado no Matlab com o tempo do mesmo algoritmo executado no Python paralelizado.
Em seguida a imagem gerada pelo Matlab foi comparada com o mapa de calor gerado pela aplicação, para assim podermos comparar a distribuição gerada pelo dois métodos.
Por último, também foi medido o tempo de resposta total da aplicação quando esta necessita processar e mostrar o resultado de uma consulta no navegador com todos os pontos contidos na base de dados usada.

## 5.1 Resultados e Comparações

Foram realizados 30 testes com o algoritmo KDE tanto no Python quanto no Matlab e em seguida foram calculados o tempo médio em segundos e o desvio padrão.

### 5.1.1 Desempenho em máquina local

Neste comparativo os algoritmos foram executados em uma máquina local com as seguintes configurações:

Processador: AMD FX 6300 Six-Core;

Memória: 8GB

Foi utilizado o parfor do Matlab para poder paralelizar o somatório contido no algoritmo do KDE e no python foi paralelizado a parte em que se calcula cada pdf em relação ao todo.

Foram usadas 14027 coordenadas de duas dimensões, latitude e longitude.

Na tabela 5.1 é apresentado o resultado comparativo entre o KDE executado no Python (sequencial e parcialmente paralelo) e no Matlab.

Tabela 5.1: KDE: Comparativo entre Matlab e Python (14027 pontos)

| Linguagem | Tempo Médio | Desvio Padrão |
|---|---|---|
| Matlab(paralelo) | 9.166467 | 0,455777031 |
| Python(paralelo) | 13.365159 | 0,591963181 |
| Python(sequencial) | 42.356532 | 0,782365834 |

As figura 5.1, 5.2 e 5.3 mostram os resultados gráficos dos algoritmos no Matlab e na aplicação respectivamente aplicados na primeira base de testes com 570 pontos, contendo a localização das agências do IBGE pelo país.

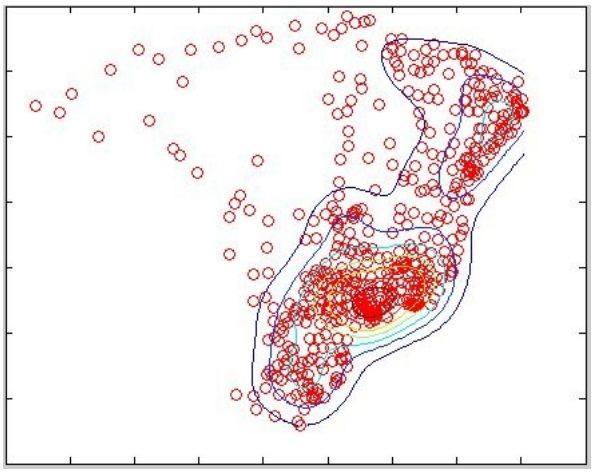


Figura 5.1: Imagem plotada pelo Matlab: IBGE, 570 pontos

Já as figuras 5.4, 5.5 e 5.6 mostram os resultados gráficos dos algoritmos executados pelo Matlab e pela aplicação respectivamente na base principal, contendo as 14027 coordenadas dos alunos da UFRRJ.

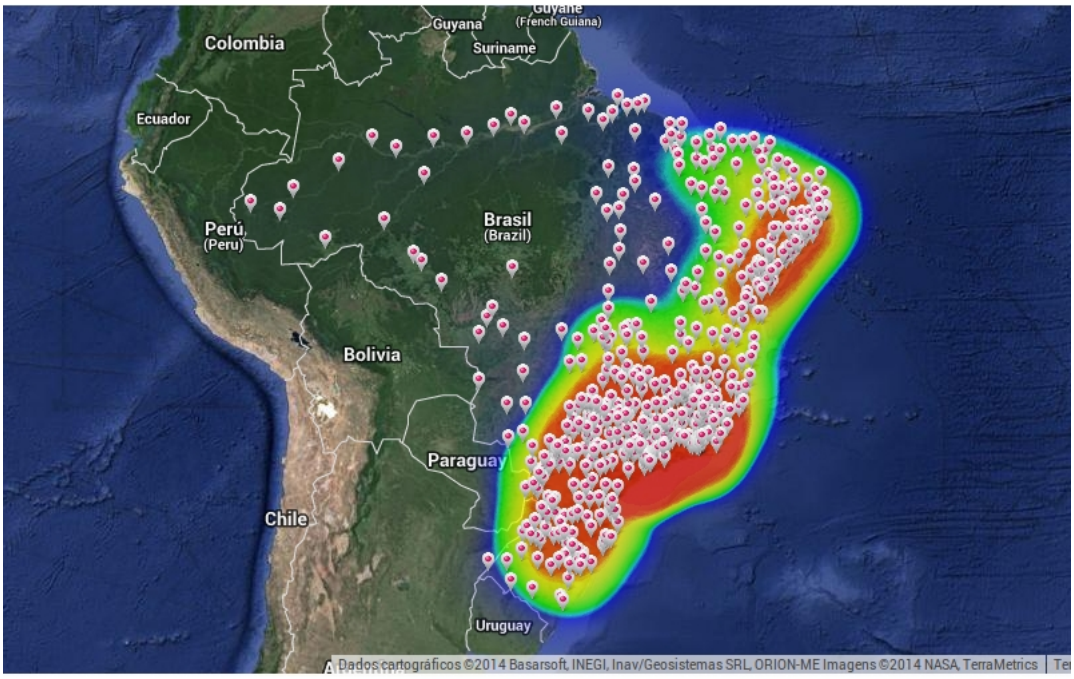



Figura 5.2: Imagem plotada pela Aplicação: IBGE, 570 pontos

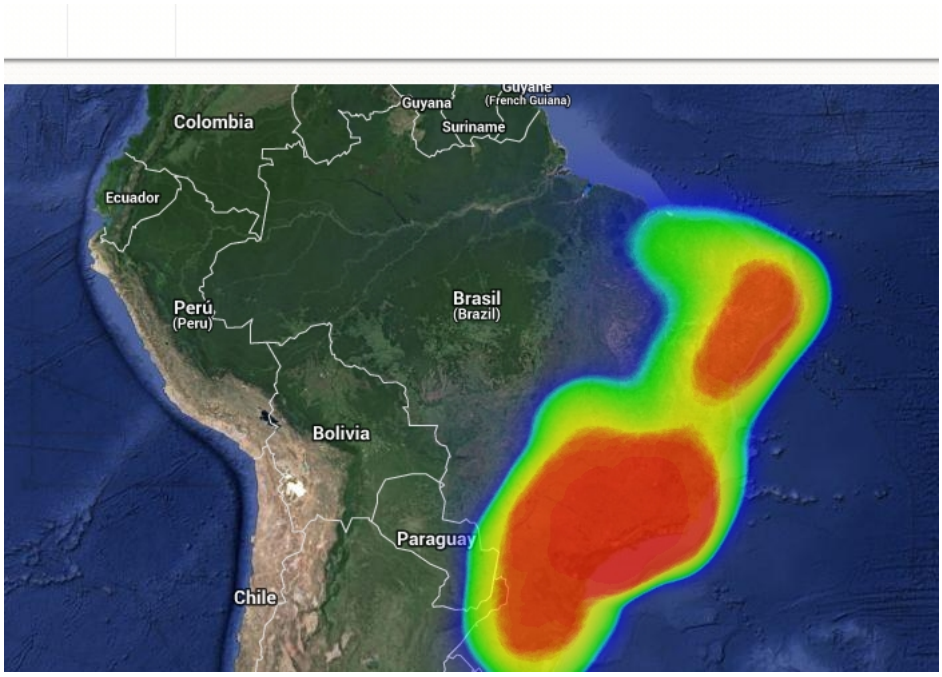



Figura 5.3: Imagem plotada pela aplicação: IBGE, 570 pontos

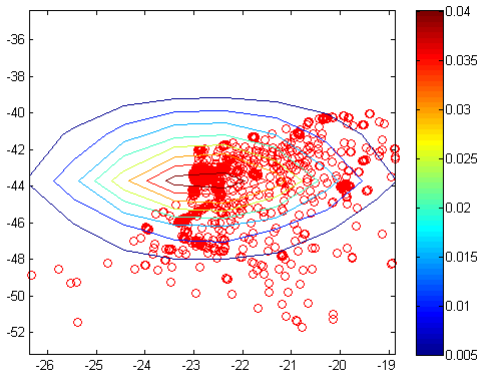



Figura 5.4: Imagem plotada pelo Matlab: Alunos da UFRRJ, 14027 pontos

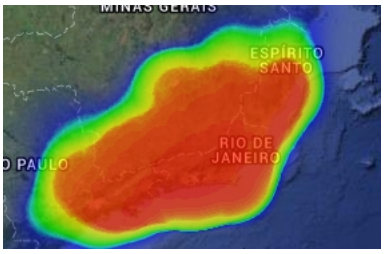



Figura 5.5: Imagem plotada pela Aplicação: Alunos da UFRRJ, 14027 pontos

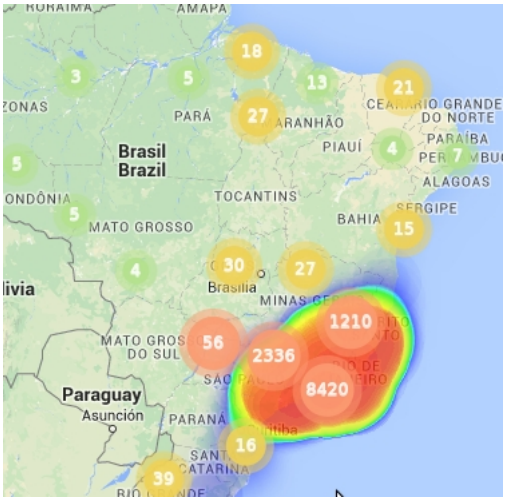



Figura 5.6: Imagem plotada pela Aplicação: Alunos da UFRRJ, 14027 pontos

### 5.1.2 Desempenho online

O servidor cloud no qual a aplicação está hospedada possui a seguinte configuração:

Apache NGinx; Processador Intel 3.0 GHz; Memória RAM de 512MB; Storage de

20GB SSD; S.O. Ubuntu 14.04.

Por conta da contratação do plano mais barato e simples de hospedagem, este servidor, em especial, possui apenas um Core lógico. Porém, como mostra a tabela 5.2, a aplicação conseguiu responder de forma razoável o processamento dos 14027 pontos.

Desta vez, o resultado mostrado leva em conta o tempo de carregamento total da aplicação depois de uma consulta, ou seja, é levado em conta o tempo desde o início da requisição até o momento em que é plotado no navegador os resultados da consulta.

Tabela 5.2: Tempo Total da aplicação (14027 pontos)

| Tempo Médio | Desvio Padrão |
|---|---|
| 17,44 (sem cache) | 0,211187 5,82 (com cache) 0,371256 |

## 5.2 Considerações Finais

Apesar da versão do Matlab ter apresentado melhores resultados, devemos levar em consideração que o KDE usado pela biblioteca Scipy neste trabalho não está o mais paralelizado possível, visto que a única parte paralelizada do algoritmo foi a parte que estima um ponto em relação aos outros.

Outro fator interessante é que 13 dos 17 segundos gastos pela aplicação, são somente para estimar as densidades, o que nos permite dizer que a aplicação demora apenas 4 segundos para fazer o resto de todo o trabalho além do processamento do KDE, ou seja, busca da informação, carregamento do mapa de calor, carregamento e clusterização dos pontos e carregamento dos gráficos.

Depois de utilizada a estratégia de cache com o Redis o tempo de carregamento dos PDFs caiu de 13 para 2 segundos e o carregamento total da página caiu em para 5 segundos ao se realizar alguma consulta pela segunda vez.

# Capítulo 6

# Conclusão

Através deste trabalho conseguimos integrar um método estimador de densidades com uma aplicação web de análise de dados e mostrar que é possível conseguir resultados parecidos com o de softwares reconhecidos pelo mercado.

Vimos que a ferramenta pode ter problemas de carregamento no browser caso se utilize milhares de pontos, devido as limitações do próprio browser mas demos uma alternativa promissora pra amenizar este problema, no caso, a clusterização e rasterização dos pontos, técnica usada nesta ferramenta.

Também foi mostrado que a velocidade de processamento do estimador é crucial para a rapidez de resposta da ferramenta e neste trabalho, apesar de não estarmos fazendo uso de GPU, conseguimos paralelizar uma parte do algoritmo e atingir um ganho interessante de aproximadamente 325% sobre o modelo sequencial.

Além disso, foi utilizada uma estratégia de cache utilizando Redis e evitamos reprocessamento desnecessário dos PDFs, fazendo com que consultas repetidas tivessem um tempo de resposta de poucos segundos, em alguns casos até menos de 1 segundo.

Portanto, concluímos que a ferramenta conseguiu atingir seu objetivo inicial que era proporcionar a análise dos dados geográficos, não só limitada a estes, de forma intuitiva, com qualidade e com boa velocidade de resposta.

## 6.1 Trabalhos futuros

Podemos elencar algumas soluções que fariam a aplicação responder de forma mais rápida, por exemplo, explorar o paralelismo no algoritmo do KDE dentro da biblioteca do SciPy. Assim o primeiro processamento do KDE seria mais rápido do que o atual.

Fazer uso de processamento paralelo em placas gráficas para calcular o KDE Multidimensional também gera bons resultados, inclusive, em um trabalho realizado anteriormente em classe, conseguimos alcançar desempenhos melhores do que o próprio Matlab somente utilizando paralelismo com o auxílio do CUDA.

Uma servidor com mais cores também aumentaria consideravelmente o processamento da aplicação. A utilização de bancos de dados distribuídos também pode ser uma boa opção para um dataset com milhões de pontos.

De posse de todas essas possibilidades podemos dizer que como trabalho futuro seria o estudo e a experimentação das tecnologias e soluções mencionadas acima combinadas entre si, na esperança de poder melhorar consideravelmente a ferramenta.

# Referências


[1] DUDA, Richard O. HART, P. E. S. D. G. Pattern classification. *John Wiley and Sons* (2012).

[2] BISHOP, C. M. Pattern recognition and machine learning. *New York: Springer* (2007).

[3] BOLSTAD, P. Gis fundamentals: a first text on geographic information. *USA, Eider Press* (2005).

[4] GUTING, R. H. An introduction to spatial database systems. *VLDB Journal 3*, 4 (1994), 357–399.

[5] MELLO, C. Agrupamento de regiões: Uma abordagem utilizando acessibilidade. *M.Sc. Thesis, Rio de Janeiro, Brazil*.

[6] Sampaio, G. B., Gazola, A., & Lisboa Filho, J Modelagem e projeto de bancos de dados geográficos com características temporais. *SIMPÓSIO MINEIRO DE SISTEMAS DE INFORMAÇÃO* (2005).

[7] Chart.js. Disponível em http://www.chartjs.org/. *acessado em Novembro* (2014).

[8] CSS3. Disponível em http://www.w3.org/tr/css/. *acessado em Fevereiro* (2014).

[9] Google Maps v.2 Disponível em https://developers.google.com/maps/documentation/javascr
br. *acessado em Outubro* (2014).

[10] KDE em Python Disponível em https://github.com/scipy/scipy/blob/master/scipy/stats/kde *acessado em Dezembro* (2014).

[11] HTML5. Disponível em http://www.w3.org/tr/html5/. *acessado em Fevereiro* (2014).

[12] Javascript. Disponível em http://www.ecma-international.org/ecma-262/5.1/. *acessado em Março* (2014).

[13] jQuery v.2.8.1. Disponível em http://jquery.com/. *acessado em Dezembro* (2014).

[14] jQueryUI. Disponível em http://jqueryui.com/. *acessado em Dezembro* (2014).

[15] Matlab. Disponível em http://www.mathworks.com/products/matlab/. *acessado em Novembro* (2014).

[16] NumPy. Disponível em http://www.numpy.org/. *acessado em Novembro* (2014).

[17] Apache Server Disponível em http://www.apache.org/. *acessado em Julho* (2014).

[18] Joblib v.0.8.4 Disponível em https://pypi.python.org/pypi/joblib. *acessado em Dezembro* (2014).

[19] Scipy v.1.9.1 Disponível em http://www.scipy.org/. *acessado em Novembro* (2014).

[20] Heatmap.js v.2 Disponível em http://www.patrick-wied.at/static/heatmapjs/. *acessado em Junho* (2014).

[21] OpenLayers.js v.2 Disponível em http://openlayers.org/. *acessado em Setembro* (2014).

[22] PostGIS v.2.1 Disponível em http://postgis.net/stuff/postgis-2.1.pdf. *acessado em Janeiro* (2014).

[23] Python v.2.7 Disponível em https://www.python.org/. *acessado em Outubro* (2014).

[24] Bootstrap.js v.3.3.1 Disponível em http://getbootstrap.com/. *acessado em Dezembro* (2014).

[25] PHP v.5.5 Disponível em http://php.net/. *acessado em Março* (2014).

[26] Postgres v.8.2 Disponível em http://www.postgresql.org/download/. *acessado em Janeiro* (2008).

[27] REDIS. Disponível em http://redis.io/. *acessado em Dezembro* (2014).