UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

LUCAS HAUPTMANN DE ALMEIDA

SISTEMA PARA ANÁLISE E PREVISÃO ESTATÍSTICA DE INDICADORES AMBIENTAIS DE POLUIÇÃO DO AR NO CONTEXTO DE *BIG DATA*

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA

2014

LUCAS HAUPTMANN DE ALMEIDA

SISTEMA PARA ANÁLISE E PREVISÃO ESTATÍSTICA DE INDICADORES AMBIENTAIS DE POLUIÇÃO DO AR NO CONTEXTO DE *BIG DATA*

Trabalho de Conclusão de Curso apresentado ao Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de "Bacharel" – Área de Concentração: Bacharelado em Sistemas de Informação.

Orientador: Etéocles da Silva Cavalcanti
Prof. Adjunto - Depto. Informática
UTFPR

Co-orientador: Jesse Thé
Diretor - *Lakes IT Group*

CURITIBA

2014

Aos meus avós, que sempre fizeram de tudo pela minha educação.

À minha mãe, que sempre me inspirou a ser uma pessoa dedicada.

À minha irmã, que sempre me incentivou a dar um passo à frente.

AGRADECIMENTOS

Agradeço a todos os professores que a vida me ofereceu, pois este trabalho é fruto da dedicação de cada um desses profissionais. Agradeço especialmente aos professores do Departamento Acadêmico de Informática da UTFPR que trabalham todos os dias para que os seus alunos se tornem profissionais críticos e aptos para enfrentar desafios nas mais diversas áreas da computação. Agradeço ao Professor Etéocles da Silva Cavalcanti que diversas vezes me mostrou o caminho correto para a conclusão deste trabalho.

Agradeço também a todos os profissionais que me ofereceram seu tempo e seu conhecimento, permitindo que eu me tornasse um profissional também. Agradeço a todos que trabalham na *Lakes Environmental*, principalmente ao Doutor Jesse Thé, que me ofereceu os recursos e o tempo necessário para desenvolver este trabalho, mostrando-me o que é Big Data e a importância da manipulação de dados, além de abrir as portas para o início da minha carreira como profissional.

Agradeço a todos os meus amigos e companheiros de curso, que durante muito tempo estiveram ao meu lado, colaborando para alcançar novos conhecimentos e obter êxito nas mais diversas disciplinas.

E por fim, agradeço a minha família, em especial aos meus avós, à minha mãe e à minha irmã, os quais sempre serão fundamentais na minha vida para vencer os mais diversos desafios.

The future belongs to the companies and people that turn data into products. (LOUKIDES, 2010).

O futuro pertence às companhias e às pessoas que transformam dados em produtos. (LOUKIDES, 2010).

RESUMO

ALMEIDA, Lucas. SISTEMA PARA ANÁLISE E PREVISÃO ESTATÍSTICA DE INDICADORES AMBIENTAIS DE POLUIÇÃO DO AR NO CONTEXTO DE *BIG DATA*. 105 f. Trabalho de Conclusão de Curso – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2014.

A constante evolução da computação e da capacidade computacional dos dispositivos de *hardware* permite que mais dados sejam armazenados e utilizados das mais diversas formas. No entanto, apesar da diversidade, aplicações desta natureza apresentam o mesmo objetivo: agregar valor a partir dos dados disponíveis, processando-os e gerando novos dados mais ricos em significado. Este novo processo é chamado de *Big Data*. Nas ciências atmosféricas, a *Big Data* está presente para suprir diferentes necessidades. Entre elas, encontra-se a necessidade de analisar e prever dados de poluição do ar. Neste contexto, este trabalho tem como objetivo apresentar um sistema para Big Data capaz de recuperar dados de forma eficiente, utilizando-os em métodos estatísticos de análise e previsão. Os dados foram obtidos da base de dados da Agência de Proteção Ambiental dos Estados Unidos, em seguida foram armazenados em um banco de dados NoSQL, que permitiu a utilização eficiente. Os métodos de análise e previsão foram desenvolvidos na linguagem de programação R, a qual facilita o uso de técnicas e modelos de séries temporais. A versão final do sistema apresenta uma interface web para acesso às análises. Ao final, são apresentados estudos de casos para demonstrar as possíveis análises que podem ser realizadas utilizando o sistema. Também são apresentados sugestões para trabalhos futuros afim de continuar a pesquisa iniciada neste documento.

Palavras-chave: Big Data, Dados, Análise Estatística, Previsão

ABSTRACT

ALMEIDA, Lucas. STATISTICAL ANALYSIS AND FORECAST SYSTEM FOR ENVIRONMENTAL INDICATORS OF AIR POLLUTION IN THE BIG DATA CONTEXT. 105 f. Trabalho de Conclusão de Curso – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2014.

The constant evolution of computing and computational capabilities of hardware devices allows more data to be stored and used in several different ways. However, despite this diversity, all applications share the same objective: to be able to obtain value from available data, processing it and generating more meaningful new data. This new process is called Big Data. In the atmospherical science field, the Big Data is present to feed different needs. Among these needs there is the need to analyze and forecast air pollution data. In this context, this document presents a Big Data system able to efficiently retrieve data, using it in different statistical analysis and forecast methods. The data was obtained from the United States Environmental Protection Agency website, then it was stored in a NoSQL database, which allows the efficient utilization. The analysis and forecast methods were developed using the programming language R, which facilitates the use of time series techniques and models. The final version of the system has a web interface to access the analyses. At the end, case studies are presented to demonstrate possible analyzes that can be executed in the system. Also, suggestions for future work are presented in order to continue the research that begins in this document.

Keywords: Big Data, Data, Statistical Analysis, Forecast

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AQS	*Air Quality System*
CSV	*Comma Separated Values*
EPA	*Environmental Protection Agency*
KIST	*Korean Institute of Science and Technology Information*
MVC	*Model-View-Controller*
NERC	*Natural Environment Research Council*
SVEC	*Structural Vector Error Correction Model*
UML	*Unified Modeling Language*
USEPA	*United States Environmental Protection Agency*
VAR	*Vector Autoregressive Model*

SUMÁRIO

1 INTRODUÇÃO

A aplicação da computação nas mais diversas áreas do conhecimento sempre foi associada ao processamento de dados. Com o crescimento do poder computacional, tornou-se possível aplicar a computação para a extração de informações de base de dados cada vez maiores. Estas aplicações eram impraticáveis há alguns anos atrás devido ao alto custo e a indisponibilidade de sistemas de armazenamento, bem como a dificuldade em organizar processadores em paralelo.

Aplicações computacionais tradicionais até hoje são desenvolvidas para realizar operações de processamento de dados. Seus recursos são organizados de forma semelhante: elas possuem uma interface de usuário e uma *middleware* para enviar e receber dados de diferentes serviços e fontes de dados. Estas aplicações, apesar de utilizarem dados, não adquirem valor a partir dos dados presentes, apenas os movimentam entre os diferentes componentes do sistema.

Loukides (2010) define um produto de dados como uma aplicação computacional que adquire valor a partir dos dados disponíveis, gerando outros dados como resultado do seu processamento. Estes dados resultantes são mais significativos do que os dados originais e podem ser apresentados visualmente em diferentes gráficos, tendências e até mesmo infográficos. Os resultados também são acessíveis para pessoas que não estão acostumadas com a análise de grandes conjuntos de valores numéricos e abstratos. Muitas vezes as informações só serão obtidas após uma análise específica e especializada dado o caráter multi-disciplinar do produto.

A característica fundamental destes produtos é a capacidade que apresentam para processar uma quantidade muito grande de dados. Sendo assim, ainda segundo Loukides, esta quantidade acaba se tornando parte do problema. Hoje, estas quantidades se encontram comumente entre 1000^3KB (*gigabytes*) e 1000^5KB (*petabytes*). No entanto estes valores, segundo observações históricas, continuam a crescer e, no futuro, o que é considerado uma quantidade "grande" será considerado uma quantidade "média". Neste contexto, para viabilizar o processamento e agregar valor, é necessário realizar a decomposição de grandes problemas

em pequenas partes, tornando possível que os resultados sejam oferecidos para os usuários de forma eficiente (PATIL, 2012).

É neste contexto que surge a *Big Data*. Este campo de estudo relativamente novo é caracterizado pelo desenvolvimento de produtos para coletar, identificar e analisar dados de forma eficiente. Este processo oferece meios para compressão dos dados, criação de informações e, consequentemente, obtenção de valores para os interessados. Este novo tipo de produto tem despertado o interesse de companhias de software, como a *Microsoft*, o *Google* e o *Facebook*, as quais têm desenvolvido aplicações neste nicho.

O *Google* desenvolveu o *Bigtable* (CHANG et al., 2008), um sistema de armazenamento que opera sobre um modelo de dados próprio (GHEMAWAT et al., 2003) para permitir que uma grande quantidade de dados sejam arquivados em vários servidores distribuídos. Com um problema parecido, o *Facebook* desenvolveu o *Cassandra* (LAKSHMAN; MALIK, 2010), um sistema de armazenamento similar ao *Bigtable*.

Apesar das definições destes sistemas serem parecidas com descrições de sistemas tradicionais de banco de dados distribuídos, os novos sistemas de armazenamento desenvolvidos no contexto da *Big Data* não apresentam propriedades relacionais provenientes da álgebra relacional, como operações de união e interseção.

O desenvolvimento de tais aplicações evidencia os novos caminhos da computação e a necessidade da criação de sistemas capazes de trabalhar com uma grande quantidade de dados. As aplicações e seus impactos vão muito além do desenvolvimento de artefatos computacionais. Hoje, a *Big Data* está presente em diversas áreas, como no sequenciamento do genoma (MARX, 2013), no sistema de saúde (MURDOCH; DETSKY, 2013), em pesquisas sobre o HIV (BUSHMAN et al., 2013) e nos sistemas de *Business Intelligence* (CHEN et al., 2012).

Além de estar presente nas áreas previamente citadas, conceitos de *Big Data* também são aplicados em diversas ciências do meio ambiente. Entre elas, destacam-se as aplicações para as ciências atmosféricas, as quais são descritas na próxima seção.

1.1 BIG DATA EM SISTEMAS PARA CIÊNCIAS ATMOSFÉRICAS

As aplicações para ciências atmosféricas, apesar de terem objetivos diferentes, trabalham com uma quantidade grande de dados, exigindo, por tanto, o desenvolvimento e/ou aplicação de sistemas para *Big Data*.

A *startup* OpenSignal (2013) desenvolveu um aplicativo para coletar dados atmosféricos de *smartphones* que possuem *hardware* específico, como termômetros e medidores de umidade. Este aplicativo pretende utilizar as vantagens do *crowdsourcing*, uma técnica para a obtenção de dados a partir de um grupo grande de pessoas, para adquirir uma grande quantidade de dados e, a partir deles, desenvolver sistemas para avaliar condições climáticas em áreas urbanas e aprimorar os sistemas tradicionais de previsão do tempo (JOHNSTON, 2013).

Outras companhias trabalham ao lado de agências governamentais para diminuir a perda de dados provenientes de satélites. Knapp (2013) discorre sobre satélites que têm capacidade de cobrir cada metro da Terra a cada quatro horas, gerando 60GB da dados por dia. O processamento destes dados geram 3TB de informações, as quais são capazes de representar as medições em três dimensões e em tempo real.

O governo da Coréia do Sul, para melhorar os seus sistemas de previsões meteorológicas, investiu em infra-estrutura para ser capaz de armazenar mais dados. Utilizando *hardware* da IBM, os sul coreanos são capazes de armazenar 9.3 *petabytes* de dados (HAMM, 2013). Além do *hardware*, a IBM também provê o sistema *Deep Thunder*, que utiliza os recursos de forma eficiente, através de algoritmos sofisticados para previsão meteorológicas em regiões de curta distância, como uma quadra ou algumas ruas, por exemplo (GALLAGHER, 2012).

Outro sistema que trabalha com *Big Data* é o *U-Air*, um sistema de aprendizagem semi-supervisionada para dados meteorológicos que utiliza classificadores espaciais e temporais (ZHENG et al., 2013). O sistema foi avaliado através de comparações com estações meteorológicas chinesas, mostrando-se eficiente, com mais de 80% de precisão.

Todas estas aplicações são importantes para as ciências atmosféricas. No entanto, como é possível avaliar um sistema desenvolvido para *Big Data*? Qual a probabilidade dos dados estarem corretos e os sistemas que irão utilizar estes dados não estarem propagando erros? Como filtrar os dados para não executar processamentos desnecessários? Estas questões são comuns no desenvolvimento de aplicações desta área. No desenvolvimento de sistemas para áreas ambientais e ciências do meio-ambiente não é diferente. Neste trabalho é apresentado o desenvolvimento de um sistema para *Big Data*, com a intenção de esclarecê-las.

1.2 JUSTIFICATIVA

Quando realizam operações sobre um número muito grande de dados, os sistemas, em geral, demoram para responder e, muitas vezes, atrasam o processamento de outros serviços. Isto pode frustrar os usuários, que deixarão de utilizar o sistema. Além disso, o atraso de outros processos pode ser prejudicial em vários sentidos.

Em razão da grande quantidade de dados, como é possível verificar nos exemplos apresentados, faz-se necessário, durante o desenvolvimento de um sistema no contexto de *Big Data*, garantir a eficiência e a eficácia na obtenção e utilização de grandes quantidades de dados.

Neste trabalho é desenvolvido um sistema de análise e previsão estatística de indicadores ambientais de poluição do ar. Esse trabalho conta com o apoio da empresa Lakes Environmental (2014). Esta empresa desenvolve soluções de tecnologia da informação para ciências ambientais, sendo referência mundial em *softwares* para modelagem de dispersão de partículas na atmosfera.

Os dados que são utilizados pelo sistema foram obtidos do Air Quality System (2014) (AQS), um sistema para gerenciamento de dados de poluentes e de correntes de ar externas, o qual é administrado pela Agência de Proteção Ambiental dos EUA (USEPA) (The United States Environmental Protection Agency, 2014).

Na segunda parte é desenvolvido o protótipo do sistema para análise e previsão dos dados. A intenção deste sistema é acessar os dados provenientes do AQS eficientemente, por isso o desenvolvimento é concentrado nos aspectos de *Big Data*. Pacotes de *software* para análise estatística e previsão são utilizados nesta etapa.

Esta etapa inclui todo o processo necessário para o desenvolvimento de software: análise e projeto, implementação e, por fim, verificação e validação. Apesar da aparência sequencial da descrição desta etapa, o sistema foi desenvolvido ciclicamente, executando-se cada etapa mais de uma vez.

Finalmente, na terceira parte, são apresentados os resultados da aplicação sob os dados disponíveis. Nesta etapa é avaliado o desempenho do sistema sobre os dados provenientes do AQS, bem como a qualidade dos dados de saída do sistema.

1.3 OBJETIVOS

No contexto da *Big Data* para ciências atmosféricas, este trabalho tem como objetivo o desenvolvimento de um sistema para análise e previsão de dados de poluição do ar. Para alcançá-lo, os seguintes objetivos específicos farão parte da solução final:

- Escolha dos dados que serão utilizados;
- Preparação dos dados escolhidos;
- Escolha dos métodos estatísticos e de previsão;
- Incorporação da solução em um sistema para Big Data.

1.4 ORGANIZAÇÃO DA MONOGRAFIA

Esta monografia está organizada em seis capítulos. Neste primeiro capítulo, tem-se a introdução, onde são apresentados o contexto, a justificativa e os objetivos do trabalho. No segundo capítulo é realizado o levantamento bibliográfico e apresentado o estado-da-arte em *Big Data*. No terceiro capítulo é apresentado a especificação do sistema e no quarto o desenvolvimento. Por fim, no quinto encontram-se os estudos de caso e no sexto as conclusões e considerações finais.

2 REVISÃO BIBLIOGRÁFICA

Para que seja possível realizar a análise e previsão de dados da qualidade do ar, é necessário entender o que se espera dos dados disponíveis e como eles serão utilizados. Assim, no contexto deste trabalho, é preciso entender os sistemas para *Big Data*.

Este capítulo traz uma revisão sobre o estado-da-arte dos diversos segmentos que compõem este campo do conhecimento, incluindo utilizações em pesquisas sobre ciências atmosféricas ou campos relacionados.

2.1 SISTEMAS PARA *BIG DATA*

A revisão do estado da arte de arquiteturas e plataformas para análise de dados e descoberta de conhecimento realizado por Begoli (2012) revela três pontos importantes para o desenvolvimento deste trabalho: i) a definição das terminologias utilizadas na área; ii) uma pequena pesquisa sobre a utilização de plataformas e ferramentas e; iii) as principais arquiteturas utilizadas pelos sistemas.

Por ser uma área de conhecimento nova e em rápido crescimento, a área de análise de dados e descoberta de informação possui várias terminologias para definir as suas atividades. Uma distinção importante realizada por Begoli é a distinção entre Descoberta de Informação em Dados e Análise de Dados.

A primeira, Descoberta de Informação em Dados, é uma terminologia utilizada para definir a atividade de grupos multi-disciplinares, que envolvem ciência da computação, estatística e especialistas no domínio da aplicação (economia, biologia etc, por exemplo), que buscam extrair novos conhecimentos de grandes conjuntos de dados.

Já a segunda, Análise de Dados, é uma terminologia utilizada para englobar as atividades de compreensão de dados através da utilização de recursos computacionais e estatísticos, bem como métodos de visualização. Entre estas atividades, encontram-se atividades das disciplinas de aprendizagem de máquina, análise de dados, visualização de

informação e análise visual.

Dentro destas definições, uma pequena pesquisa foi realizada para verificar quais plataformas estão sendo as mais utilizadas entre os grupos de Descoberta de Informação em Dados. Entres os *softwares* mais populares para mineração de dados, aparecem a Linguagem R, o Excel, o *Rapid-IRapid Miner*, o Weka e o Pentaho. Entre as tecnologias mais populares para *Big Data*, aparecem o *Hadoop*, o *Amazon Web Services* (AWS) e tecnologias *NoSQL*, que são alternativas aos bancos de dados tradicionais. Finalmente, entre as linguagens de programação mais utilização para o desenvolvimento de código customizado, aparecem a Linguagem R e a Linguagem Java.

Conhecer as ferramentas que estão sendo utilizadas é importante para compreender o estado atual dos artefatos que são resultados das pesquisas para Descoberta de Informação em Dados. A familiaridade com estas ferramentas é fundamental para facilitar a compreensão das pesquisas que estão sendo realizadas, o que facilitará o desenvolvimento deste trabalho.

Finalmente, Begoli aponta as principais arquiteturas por trás das ferramentas, tecnologias e linguagens de programação mais utilizadas no ano de 2012. Além dos tradicionais sistemas de bancos de dados relacionais e das plataformas de computação de alta performance, ambos mais conhecidos por serem estudados e desenvolvidos antes mesmo do termo *Big Data* ser definido, encontram-se o algoritmo *MapReduce*, o qual é utilizado para redução de dados através de operações de seleção, ordenação e síntese de dados, e as tecnologias de gerenciamento de dados *NoSQL*.

Os bancos de dados relacionais e os sistemas necessários para o seu gerenciamento, por serem utilizados há muitas décadas e, portanto, apresentarem um caráter de sistema legado, ainda que tenham sido desenvolvidos consideravelmente, estão presentes na maioria das empresas que possuem uma grande quantidade de dados, sendo este segmento o mais disputado pelo mercado.

Já os sistemas para computação de alta performance se caracterizam pela necessidade de execução em paralelo e da realocação de dados para computação, o que exige que estes sistemas apresentem diversas características como, por exemplo, grande largura de banda, baixa latência de rede, alto número de núcleos de processamento e grande capacidade de memória. Sobre todos esses recursos de *hardware*, ainda se faz necessário a utilização de *softwares* especializados.

Em geral, estudos nestas duas áreas focam no desempenho do processamento de grandes volume de dados em processadores distribuídos. Já o estudo da utilização do

algoritmo *MapReduce*, e da utilização das tecnologias *NoSQL* é desenvolvido com foco no armazenamento, recuperação e análise de dados.

A fim de explorar áreas do conhecimento, bem como ferramentas que possam ser utilizadas no desenvolvimento de sistemas customizados, as próximas seções apresentam detalhes sobre as definições e as utilizações destas novas tecnologias. Também serão apresentados detalhes sobre a Linguagem R.

2.1.1 MAPREDUCE

Dado as definições anteriores, os sistemas para *Big Data* podem ser resumidos como algoritmos, tecnologias e artefatos computacionais desenvolvidos para extrair informações de uma grande base de dados de forma dinâmica e eficiente, muitas vezes utilizando sistemas distribuídos. Nas palavras de Fisher (FISHER et al., 2012):

"Fundamentalmente, análise em *Big Data* é um fluxo de trabalho no qual são destilados terabytes de dados de pouco valor em, algumas vezes, um único bit de dados de alto valor".

Observando a necessidade por sistemas com essas características ocorrer diversas vezes e em diferentes projetos, os engenheiros do *Google*, Jeffrey Dean e Sanjay Ghemawat (DEAN; GHEMAWAT, 2012) desenvolveram o *MapReduce*.

Inspirados pelas primitivas *map* e *reduce* presentes na linguagem funcional *Lisp*, eles desenvolveram um sistema capaz de mapear (*map*) valores de entrada em pares de chave e valor, reduzindo-os (*reduce*), ao final, em uma única saída de pares com chaves em comum. Este processo possibilitou a utilização de dados que, em condições limitadas de processamento, não caberiam em memória se fossem carregados em seu formato original.

Por ser simples e apresentar tarefas bem definidas, o *MapReduce* foi projetado para funcionar paralelamente em múltiplos servidores, uns responsáveis por mapear e outros responsáveis por reduzir. Assim, o conjunto de dados que compõe a entrada é dividido antes de ser mapeado, e cada máquina designada para mapear pode processar uma parte do conjunto. Terminado o processamento, cada máquina responsável por reduzir os pares espera por n pares e os reduz a uma única saída, sendo n definido pelo usuário. Este fluxo pode ser observado na Figura 1.

Se, por exemplo, deseja-se contar quantas vezes cada palavra aparece em um dado conjunto de entrada composto por vários arquivos de texto, é possível considerar cada arquivo como uma divisão do conjunto de dados e cada divisão pode ser processada em paralelo. Durante o processamento cada palavra é utilizada como chave e a ela é atribuído o valor 1.

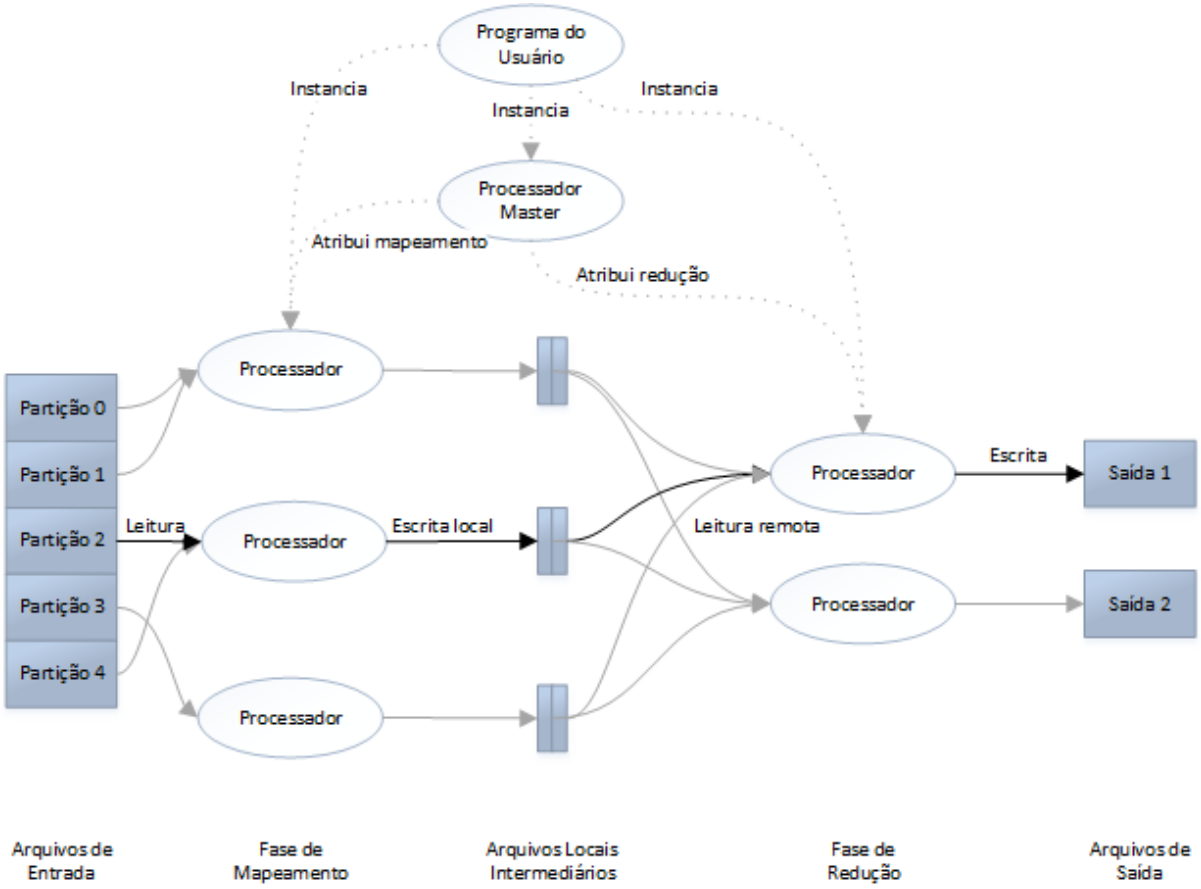

Figura 1: Fluxo de dados no Algoritmo MapReduce.

Fonte: Adaptado de Dean e Ghemawat (2012)

Isto significa que aquela palavra foi encontrada uma vez. Finalmente, após serem gerados todos os pares, reduz-se as chaves iguais em uma única saída, ou seja, se a palavra "dados" apareceu 10 vezes, tem-se 10 pares [dados — 1], os quais são reduzidos para a saída [dados — 10].

A utilização de um sistema distribuído, ainda que com tarefas bem definidas, o que é ideal para a execução em paralelo, está sujeita a falhas de máquina e a falhas de rede. Para evitar estas falhas, o *MapReduce* é implementado com precauções comum a sistemas distribuídos. No algoritmo original, um servidor principal é eleito para executar *pings* nas demais máquinas, removendo da rede máquinas que não respondem depois de um tempo determinado pelo usuário. Tarefas completas de mapeamento executadas por máquinas que foram removidas são executadas novamente, porque o resultado parcial é armazenado localmente na máquina. Já as tarefas de redução completas, executas por máquinas removidas, não são executadas novamente porque o resultado é armazenado em uma saída de dados global.

Com a intenção de tornar o processo ainda mais eficiente, alguns refinamentos são aplicados. Entre eles, tem-se: funções para dividir os dados de entrada de forma que cada

divisão tenha um número balanceado de dados; ordenação dos dados para facilitar o acesso e melhorar a apresentação dos dados; combinação de chaves em processos intermediários, caso existam, utilizando a mesma função de redução; e descarte de dados que causem falhas na execução.

Todas essas características tornaram o *MapReduce* um algoritmo adotado por vários pesquisadores, até mesmo por aqueles com pouco conhecimento sobre o desenvolvimento de sistemas distribuídos.

A implementação mais famosa do *MapReduce* é o *Hadoop* (Hadoop, 2013). Desenvolvido pela *Apache Software Foundation*, o *Hadoop* é uma implementação *open-source* em *Java* do *MapReduce*. Por possuir o código aberto, muitas pesquisas foram realizadas com ele e muitos *plug-ins* foram desenvolvido, aumentando o número do tipo de aplicações que se beneficiam do *MapReduce* quando executam suas tarefas.

Uma das maiores aplicações que utilizam o *Hadoop* é o Yahoo! Search Webmap (2013). Utilizando mais de 10.000 núcleos de processamento, esta ferramenta processa e armazena dados sobre todas as páginas conhecidas na internet, utilizando estes dados como entrada para um sistemas de aprendizagem de máquina.

Quanto a otimização do algoritmo *MapReduce*, vários pesquisadores têm trabalhado em diferentes áreas e não somente os seus criadores. Guo, Fox e Zhou (GUO et al., 2012) realizaram estudos quanto a localização física dos dados antes de sua utilização no *MapReduce*, buscando atingir uma melhoria geral de performance durante a execução.

No Brasil, Kolberg e Geyer, do Instituto de Informática da Universidade Federal do Rio Grande do Sul (UFRGS), desenvolveram um simulador de plataformas de *MapReduce* (KOLBERG et al., 2013), chamado MRSG. O desenvolvimento deste simulador é importante para reduzir custos e esforços nos aspectos de escalabilidade, além de auxiliar na concepção de novas soluções, servindo como uma plataforma de testes antes de uma implementação real.

O MRSG utiliza o simulador *SimGrid* como base para simular um ambiente de *cluster*, ou grade, através de chamadas de funções, sem alterá-las. Assim, são implementadas as seguintes funções do *MapReduce*: distribuição de arquivos com replicação de blocos, mapeamento, redução, transferência de dados e execução especulativa.

Na área das ciências atmosféricas, o *MapReduce* foi utilizado por YunHee Kang (KANG et al., 2013) no pós-processamento de dados em um sistema do Instituto Coreano de Ciência e Tecnologia de Dados (KISTI - *Korean Institute of Science and Technology Information*). Este sistema possui um portal de acesso que agrega dados de diversas fontes

e aplicações. Através deste portal os usuários podem obter dados uniformes, contínuos e personalizados. No entanto, devido a estrutura distribuída do sistema, diversos dados devem ser movidos e processados para que o usuário obtenha o resultado desejado. Portanto, a redução dos dados se faz necessária para movimentá-los.

A execução deste sistema, com objetivo de prever a condição atmosférica através da utilização de uma simulação *Emsemble* com condição inicial múltipla, tem como resultado, para a previsão de condições atmosféricas a cada 1.800 segundos (0.5 horas) numa faixa de 100 anos, utilizando 30 membros *Emsemble*, um arquivo NetCDF de 30 TBytes, o qual é reduzido para 10 TBytes após o pós-processamento.

O arquivo NetCDF resultante, que representa pontos de observação, séries temporais, grades espaciais e imagens de satélite ou radar, tinha inicialmente 2,942 Mbyte e 79 variáveis. Após agrupar 5 variáveis em um subconjunto, o tempo de acesso é reduzido em 276%, de 4.10 segundos para 1.40 segundos.

A utilização do *MapReduce* garante que os dados sejam reduzidos. Para exemplificar, o autor apresenta o conjunto de dados de precipitação global por ano. Após a redução dos dados, estes são encaminhados para um sistema de análise estatística.

A importância do *MapReduce* está associada diretamente às pesquisas nas quais ele é utilizado, tanto na indústria, como na academia, contribuindo para a definição de *Big Data* como um área com necessidades diferentes da área que estuda os bancos de dados tradicionais. Neste contexto, outras alternativas foram propostas quanto a utilização de sistemas que não dependessem da utilização de bancos de dados relacionais. Entre elas estão as tecnologias *NoSQL*.

2.1.2 NOSQL

Com a utilização de tecnologias se tornando cada vez mais ubíqua, é exigido o desenvolvimento de novas formas de computação. Dentre as várias necessidades, uma das mais críticas é a necessidade de armazenar e recuperar dados eficientemente, como aponta Seltzer (SELTZER, 2008), o que reforça a ideia da utilização do *MapReduce* apresentada anteriormente para recuperar dados e abre espaço para analisar as formas de armazenamento.

Na década de 1970, com o advento da TI, as aplicações de processamento de dados para empresas ficaram em alta. A demanda por estes *softwares* exigiu que os dados que seriam utilizados fossem normalizados em tabelas para estruturar a forma de manipulação. Com isso, durante os 30 anos seguintes, esta foi a forma adotada para a manutenção de dados nas mais

diversas aplicações.

A adoção cada vez mais comum da TI permitiu que as décadas seguintes fossem marcadas pelo rápido desenvolvimento das capacidades de *hardware*. Os processadores ficaram mais rápidos e a capacidade de armazenamento dos discos rígidos ficaram maiores. Porém, a comunicação entre as memórias e as unidades de armazenamento não mudaram no mesmo ritmo e a arquitetura utilizada em computadores de pequeno e médio porte para o armazenamento de dados continuou igual àquela utilizada na década de 1970. Deste modo, muitos sistemas ainda não estão preparados para trabalhar com um grande volume de dados, porque não recebem atualizações. Apenas sistemas utilizados por organizações com esta necessidade específica foram adaptados, como é o caso dos *mainframes* (Gain vital insight from your data, 2013).

Em 2000, Chaudhuri e Weikum (CHAUDHURI; WEIKUM, 2000) demonstraram esta obsolescência na arquitetura dos bancos de dados e apontaram sinais da crise neste campo da computação em sete aspectos principais:

- O excesso de recursos nos sistemas de banco de dados (as empresas que comercializam sistemas para banco de dados colocam novos recursos por trás de um mesmo *front-end* para manterem-se no mercado, tornando o sistema complexo. No final, cada usuário acaba utilizando uma pequena porção de um grande sistema);

- A utilização de SQL (os desenvolvedores de software acabam sendo obrigados a utilizar SQL para acessar os bancos de dados relacionais. No entanto, a grande maioria não tem um grande domínio sobre a linguagem);

- A impossibilidade de prever a performance (com a adição de novas funcionalidades e o aumento da complexidade dos bancos de dados, é impossível prever a performance das operações de consulta);

- A dificuldade em ajustar os sistemas (assim como é impossível prever a performance dos bancos de dados, é difícil ajustar todos os parâmetros para cada tipo de aplicação para se obter um resultado eficiente. Empresas acabam sendo responsáveis por contratar especialistas ou por gastar mais tempo em processos de tentativa e erro até obterem um bom resultado);

- A sobrecarga de funções (muitos outros serviços, como servidores, possuem funções de armazenamentos próprias que foram originadas em sistemas de banco de dados. Neste caso a utilização de um banco de dados em conjunto deve ser cuidadosa para se evitar conflitos e repetição de funções);

- As consequências da utilização dos sistemas (bancos de dados tradicionais não estão preocupados com os novos ambientes que precisam utilizar sistemas de armazenamento, como os sistemas para dispositivos móveis que precisam somente de algumas funções específicas, portanto, não estão preocupados com as consequências dos sistemas nestes ambientes);
- A frustração nas pesquisas em banco de dados orientados a sistemas (é cada vez mais difícil realizar pesquisas em bancos de dados fora de uma companhia deste mercado devido a alta necessidade de prototipação e experimentação).

Mais recentemente, estas falhas no modelo tradicional de bancos de dados e distanciamento das aplicações foram reforçadas por Michael Stonebraker. Em seu primeiro artigo sobre o assunto, Stonebraker e Cetintemel (2005) apontam a necessidade de arquiteturas diferentes para serviços diferentes, reforçando que um tamanho não serve para todos.

Utilizando exemplos de sistemas para os quais os tradicionais banco de dados relacionais não são uma escolha apropriada, Stonebraker destaca os sistemas para *data warehouse* e os sistemas para o processamento de *streaming*.

Sistemas para *data warehouses* são utilizados primeiramente para consultas *ad-hoc*, as quais geralmente são complexas e tem por objetivo encontrar tendências históricas para análises de *business inteligence*, enquanto os sistemas tradicionais têm como objetivo atualizar campos em tabelas. Esta diferença faz com que os fabricantes de sistemas de bancos de dados relacionais criem módulos para aplicações de *data warehouse* sob um *parser* comum para todas as aplicações. Apesar do resultado final ser um *front-end* comum para os dois tipos de aplicação, virtualmente o resultado é diferente e a performance é pior para o processamento das consultas *ad-hoc*.

Já no caso dos sistemas para o processamento de *streaming*, os quais estão se tornando mais comuns devido a maior utilização de sensores em rede como, por exemplo, para o monitoramento de veículos de uma frota em tempo real, o problema na utilização da arquitetura tradicional para banco da dados está no atraso causado pela necessidade de escrever o dado no banco de dados antes que este esteja pronto para a utilização pelo sistema. Em sistemas de *streaming* os dados que seriam escritos são colocados em memória quando chegam dos sensores e em seguida enviados para a aplicação onde serão processados. Opcionalmente, estes dados podem ser escritos em paralelo ao seu processamento.

Estes dois primeiros exemplos foram reforçados através de mais três contribuições realizadas por Stonebraker et al. (2007). Neste segundo artigo os autores explicam o

problema da utilização de bancos de dados relacionais no mercado de processamento de textos. Em seguida realizam uma comparação entre arquiteturas especializadas e as arquiteturas tradicionais atualmente presentes no mercado.

Em conclusão aos seus trabalhos anteriores e como início de uma nova era arquitetural, Stonebraker resume os detalhes dos seus trabalhos na área e propõem uma arquitetura para o desenvolvimento do sistema *H-Store* para o processamento de transações *online* (STONEBRAKER et al., 2007).

A discussão criada pelos autores fomentou a busca pelo detalhamento das falhas existentes nos sistemas atuais, bem como a busca por tecnologias específicas para cada tipo de aplicação.

Para Seltzer (2008), as novas tecnologias presentes no cotidiano atual que destacam-se por exigirem novas arquiteturas são os sistemas para *warehousing*, os sistemas de diretórios de serviços, os sistemas para *web search*, os sistemas de *caching* para dispositivos móveis e os sistemas para gerenciamento de documentos XML.

Nas aplicações para ciências atmosféricas, muitos dados são processados em tempo real ao mesmo tempo em que muitos dados são recuperados de séries históricas para alimentarem sistemas de aprendizagem de máquina, sendo o alinhamento destes dois tipos de dados necessário para estudos na área.

Séries temporais constituem uma das maiores fontes de dados e necessidades de sistemas de *Big Data* para o seu processamento. Na busca pelo aprimoramento do armazenamento e do acesso a estes dados, Shafer et al. (2013) apontam as principais necessidades deste tipo específico de dados. A primeira é a necessidade de sistemas capazes de acessar uma extensão de dados de uma só vez, a segunda é a necessidade de realizar um pré-processamento nos dados para que eles sejam resumidos, e a terceira é a necessidade de comprimir os dados.

Após o levantamento do estado da arte dos sistemas utilizados para análise de séries temporais, os autores apontam que um sistema ideal deve apresentar três características: possuir uma base de dados focada em em séries temporais, suportar o armazenamento de dados de multiresolução e suportar operações sobre faixas de *stream*. Avaliando-se sistemas utilizados no mercado, os que chegam mais próximos a este modelo ideal são os sistemas de armazenamento em colunas. Isto posto, os autores propõem uma arquitetura para redução do espaço físico necessário para armazenar dados de séries temporais a fim de aumentar ainda mais a eficiência destes sistemas.

2.1.3 LINGUAGEM R

Ainda segundo o levantamento realizado por Begoli (BEGOLI, 2012), a linguagem de programação mais utilizada no ano de 2012 no desenvolvimento dos sistemas *Big Data* foi a Linguagem R (The R Project for Statistical Computing, 2014).

Esta linguagem, e também ambiente, para computação estatística é um projeto GNU criado por Ross Ihaka e Robert Gentleman. Considerada como uma implementação diferente da linguagem S (linguagem usada anteriormente à R para computação estatística), a linguagem R oferece uma ampla variedade de métodos estatísticos e gráficos, a qual pode ser estendida para dar suporte a outros tipos de cálculos necessários para aplicações específicas.

Estas características permitiram a aceitação da linguagem R e muitas aplicações foram desenvolvidas nas mais diversas áreas.

Sendo uma linguagem de programação para estatística, muitos pacotes são desenvolvidos para manipular diferentes formatos de dados. Michna e Woods (2013) desenvolveram o RNetCDF, um pacote que permite que programas escritos em R sejam capazes de manipular arquivos NetCDF, os quais são muito utilizados em simulações atmosféricas, como já citado anteriormente.

Já Shang (2013) criou o pacote *ftsa* para modelar e prever séries temporais. Os cálculos são feitos através de funções de regressão de componentes principais, permitindo que um grande número de variáveis seja reduzido, sendo possível extrair padrões dos conjuntos de entrada.

Sax e Steiner (2013) também trabalharam com séries temporais, para as quais desenvolveram o pacote temodisagg, o qual permite desagregar valores das suas unidades temporais, alterando-as para unidades temporais maiores ou menores, como, por exemplo, transformar uma série anual em uma série mensal.

Além desses pacotes, os quais não foram desenvolvidos especificamente para as ciências atmosféricas, mas que podem ser utilizados caso necessário, o pacote OpenAir (OpenAir, 2014) foi criado.

Este pacote é um projeto de intercâmbio de conhecimento do Conselho de Pesquisa do Meio-Ambiente, do inglês *Natural Environment Research Council* (2014), liderado pelo Grupo de Pesquisa em Meio-Ambiente, do inglês *Environmental Research Group*, do *King's College London* com apoio da Universidade de Leeds.

A intenção dos autores, ao desenvolverem este pacote, é superar barreiras comuns na análise de dados de poluição do ar, presentes principalmente pelo grande número de dados

disponíveis, como quais análises devem ser feitas, qual o custo associado e quais softwares podem ser utilizados (CARSLAW; ROPKINS, 2012). Dentre as funções presentes neste pacote, encontram-se funções para gerar gráficos de rosa dos ventos, rosa de poluição, outros gráficos polares, séries temporais, gráfico de calendário, gráfico de dispersão, relação linear, dentre outras.

2.1.4 FECHAMENTO

Com as definições apresentadas neste capítulo, as quais formam as bases para o desenvolvimento deste trabalho, é possível compreender o sistema que será desenvolvido para alcançar os objetivos propostos. No próximo capítulo, são apresentados os componentes da arquitetura do sistema e outros elementos que o definem quanto à sua estrutura, comportamento e interação.

3 ESPECIFICAÇÃO DO SISTEMA

Um sistema para *Big Data* deve apresentar três componentes principais: os dados, as tecnologias *Big Data* para obtenção e armazenamento destes dados e os componentes de processamento e encaminhamento destes dados. O diagrama apresentado na Figura 2 ilustra estes componentes. Os dados se caracterizam pela grande quantidade e não uniformidade. Já as tecnologias de *Big Data* são utilizadas para recuperar e reduzir estes dados. Por fim, é realizado o processamento sobre o conjunto de dados resultantes da etapa de *Big Data*.

Figura 2: Visão geral de um sistema big data.
Fonte: Autoria Própria

A grande diferença entre um sistema convencional e um sistema para *Big Data*, portanto, está no processo intermediário entre os dados e o processamento. Cada aplicação *Big Data* apresentará uma arquitetura distinta, porém todas possuem alguma das tecnologias apresentadas anteriormente.

No desenvolvimento do sistema *Big Data* deste trabalho, adotou-se uma metodologia de desenvolvimento incremental e espiral, como definido por Boehm (1988). Por este motivo, a próxima seção apresenta o histórico do desenvolvimento da especificação, que, na versão final deste documento, é diferente das versões parciais. No entanto, as seções dedicadas a explicar cada parte da especificação apresentam a versão mais recente.

Assim, a arquitetura do sistema e os blocos que foram desenvolvidos são apresentados após os histórico. A especificação ocorre através de descrições e de diagramas UML.

3.1 HISTÓRICO DE ESPECIFICAÇÃO

Tabela 1: Histórico de Especificação

Data	Versão	Autor	Descrição
23/03/2014	1	Lucas H. de Almeida	Arquitetura inicial do sistema
05/05/2014	2	Lucas H. de Almeida	Casos de uso e requisitos
17/05/2014	3	Lucas H. de Almeida	Visão modular
08/06/2014	4	Lucas H. de Almeida	Importação dos dados
27/06/2014	5	Lucas H. de Almeida	Leitura e processamento dos dados
04/07/2014	6	Lucas H. de Almeida	Interface web para acesso

Fonte: Autoria própria.

3.2 ARQUITETURA DO SISTEMA

O sistema contém quatro componentes principais, os quais podem ser considerados subsistemas, sendo, portanto, independentes uns dos outros, o que torna possível a sua utilização no futuro, como subsistemas de outros sistemas maiores. A visão geral destes três componentes é apresentada na Figura 3. Cada subsistema apresenta uma cor diferente no diagrama.

É possível, por tanto, identificar cada uma das partes essenciais de um sistema *Big Data* apresentados na Figura 2 neste diagrama. A obtenção dos dados será realizado por um *web crawler* que alimentará uma base de dados *MongoDB*, onde residem as tecnologias *Big Data* que serão utilizadas. O *MongoDB* foi escolhido por ser um banco de dados NoSQL e *open-source* escrito em C++. Essa base de dados será lida pela interface web, a qual se comunicará com um servidor para que este realize o processamento dos dados. Por fim, o resultado será entregue para o usuário especialista através do mesmo servidor web.

Figura 3: Arquitetura do Sistema

Fonte: Autoria Própria

3.2.1 REQUISITOS DO SISTEMA

A definição do sistema permite mudar a perspectiva de desenvolvimento. Sabendo quais partes estarão presentes na versão final do *software*, pode-se pensar em quais funcionalidades serão necessárias. Nesta seção, os requisitos funcionais são apresentados e, para cada requisito funcional, são listados os requisitos não-funcionais que estão imediatamente relacionados, justificando-se melhor as suas escolhas.

Os requisitos funcionais do sistema tem como objetivo caracterizar o seu funcionamento e a interação dos usuários, sendo fundamental para a criação e interpretação dos casos de uso do sistema. Já os requisitos não-funcionais são necessidades em termos não tangíveis ao usuário, mas fundamentais para que o software tenha qualidade.

Além disto, a definição de quais funções serão implementadas permite que, ao final do desenvolvimento, seja possível quantificar o que foi desenvolvido, estabelecendo o quão completo o sistema é, e quais estágios de desenvolvimento serão necessários no futuro, afim de que o *software* seja um produto completo. As Tabelas 2 a 10 descrevem os requisitos.

Tabela 2: Requisito Funcional 1

Nome: Acessar os dados da base de dados
Descrição: O usuário deverá ser capaz de acessar os dados disponíveis na base de dados
Requisitos não funcionais:
Capacidade: o sistema deve oferecer métodos para mover os dados para memória
Confiança: os dados acessados devem ser exatamente os dados solicitados pelo usuário
Disponibilidade: os dados devem estar sempre disponíveis
Escalabilidade: o sistema deve suportar a necessidade de crescimento
Extensibilidade: o sistema deve permitir que novas bases de dados sejam utilizadas no futuro
Performance: a interação com os dados deve ser eficiente

Tabela 3: Requisito Funcional 2

Nome: Escolher métodos para análise ou previsão de séries temporais
Descrição: O usuário deverá ser capaz de escolher os métodos que deseja executar
Requisitos não funcionais:
Expansão: o sistema deve oferecer meios para a adição de novos métodos
Configurações: o usuário deve ser capaz de configurar os métodos escolhidos
Documentação: os métodos devem ser documentados para facilitar a utilização

Tabela 4: Requisito Funcional 3

Nome: Importar os dados
Descrição: O sistema deverá ser capaz de importar os dados automaticamente
Requisitos não funcionais:
Confiabilidade: o usuário deverá ser capaz de confiar nos dados obtidos automaticamente
Tolerância: o sistema deve ser capaz de identificar falhas na importação e se recuperar

Tabela 5: Requisito Funcional 4

Nome: Executar o método escolhido
Descrição: O usuário deverá ser capaz de executar o método escolhido
Requisitos não funcionais:
Acessibilidade: informações da execução devem ser acessíveis

Tabela 6: Requisito Funcional 5

Nome: Visualizar o resultado do método escolhido
Descrição: O usuário deverá ser capaz de visualizar o resultado do método escolhido
Requisitos não funcionais:
Acessibilidade: os resultados da execução devem ser acessíveis

Tabela 7: Requisito Funcional 6

Nome: Acessar o sistema (*login*)
Descrição: O usuário deverá ser capaz de acessar o sistema com suas credenciais
Requisitos não funcionais:
Confiança: as credenciais devem ser transmitidas de forma segura
Disponibilidade: o acesso deve estar sempre disponível

Tabela 8: Requisito Funcional 7

Nome: Encerrar o acesso (*logout*)
Descrição: O usuário deverá ser capaz de encerrar o acesso ao sistema
Requisitos não funcionais: Disponibilidade: o encerramento deve estar sempre disponível

Tabela 9: Requisito Funcional 8

Nome: Registrar um novo cadastro
Descrição: O usuário deverá ser capaz de registrar um novo cadastro no sistema
Requisitos não funcionais: Confiança: as credenciais devem ser armazenadas de forma segura Disponibilidade: o registro de um novo cadastro deve estar sempre disponível

Tabela 10: Requisito Funcional 9

Nome: Encerrar cadastro
Descrição: O usuário deverá ser capaz de encerrar o seu cadastro
Requisitos não funcionais: Disponibilidade: o encerramento de cadastros deve estar sempre disponível

3.3 CASOS DE USO

Neste ponto, após a definição de ambas as perspectivas, a do sistema, através da arquitetura, e a do usuário final, através dos requisitos funcionais, é possível visualizar o sistema em termos de casos de uso conforme ilustrado na Figura 4, identificando atores e as possíveis interações com o sistema.

Assim sendo, identifica-se que um usuário deverá se capaz de se cadastrar, acessar o sistema, fazendo *log-in*, e encerrar o acesso, fazendo *log-out*. Após o acesso, deverá ser possível escolher os dados para análise ou previsão, escolher os métodos de análise ou previsão e solicitar a execução com base no que foi escolhido. Por fim, o usuário deverá ser capaz de visualizar os resultados. Se desejar, o usuário poderá excluir o seu cadastro da base de dados. A Figura 4 apresenta estes casos de uso.

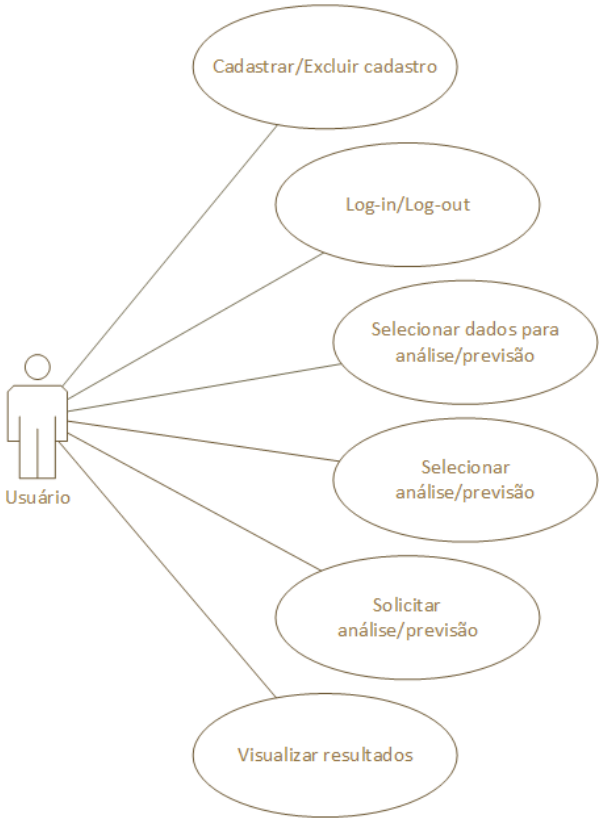

Figura 4: Diagrama de Casos de Uso

Fonte: Autoria Própria

3.4 VISÃO MODULAR

Para facilitar a compreensão e iniciar o desenvolvimento de cada um dos módulos definidos pela arquitetura do sistema, é preciso compreender a individualidade de cada módulo e quais as funcionalidades de cada um. Como os dados são provenientes do sistema AQS, esta sigla é utilizada como prefixo no nome de cada módulo.

O módulo para obtenção e armazenamento é chamado de AQS.ImportUtility. Já o módulo de acesso aos dados é chamado de AQS.MongoAccess, pois seu núcleo é formado pelo banco de dados não relacional (NoSQL) MongoDB. O módulo de processamento é chamado de AQS.R, pois a linguagem de programação R é utilizada para realização das análises e previsões. Por fim, o módulo de acesso é chamado de AQS.MVC, pois é uma interface web desenvolvida na arquitetura MVC (*Model View Controller*).

O primeiro módulo, AQS.ImportUtility, está dividido em duas partes. A primeira, chamada FileControl, é responsável por acessar a fonte de dados, copiando-os para um diretório local, verificando se eles não foram corrompidos durante a obtenção, descomprimindo-os e, por fim, excluindo-os, mantendo apenas a versão descomprimida dos dados. Na segunda parte deste mesmo módulo, chamada de ImportControl, os dados descomprimidos são serializados e preparados para inserção em lotes, a qual ocorre através da utilização do segundo módulo, AQS.MongoAccess.

O módulo AQS.MongoAccess é utilizado para interagir com o banco de dados MongoDB. A escolha e os detalhes de implementação deste módulo serão apresentados em outro capítulo. Em linhas gerais, através deste módulo é possível conectar-se ao banco de dados, preparar consultas, executá-las e deserializar os resultados.

O módulo seguinte, AQS.R, é responsável por processar os dados. Neste módulo é onde os dados são transformados, gerando mais dados ou mecanismos para interpretação dos dados existente, agregando valor aos dados e, portanto, fazendo desta aplicação um produto de dados. Neste módulo as séries temporais são escritas em arquivos individuais e scripts escritos na linguagem R são executados.

Por fim, o último módulo, AQS.MVC, oferece uma interface web através da qual os usuários podem solicitar a execução de métodos para análise ou previsão de dados. A interface também oferece suporte a todos os casos de uso, portanto, este módulo permite que o usuário acesse e encerre o acesso ao sistema, selecione dados e métodos de análise ou previsão, executando-os e visualizando os resultados obtidos.

A Figura 5 resume os métodos de cada módulo e, além disso, mostra o relacionamento entre eles. Para o usuário final, a comunicação entre os módulos será transparente, como se o sistema fosse um único módulo que executa todos os casos de uso e implementa todos os requisitos.

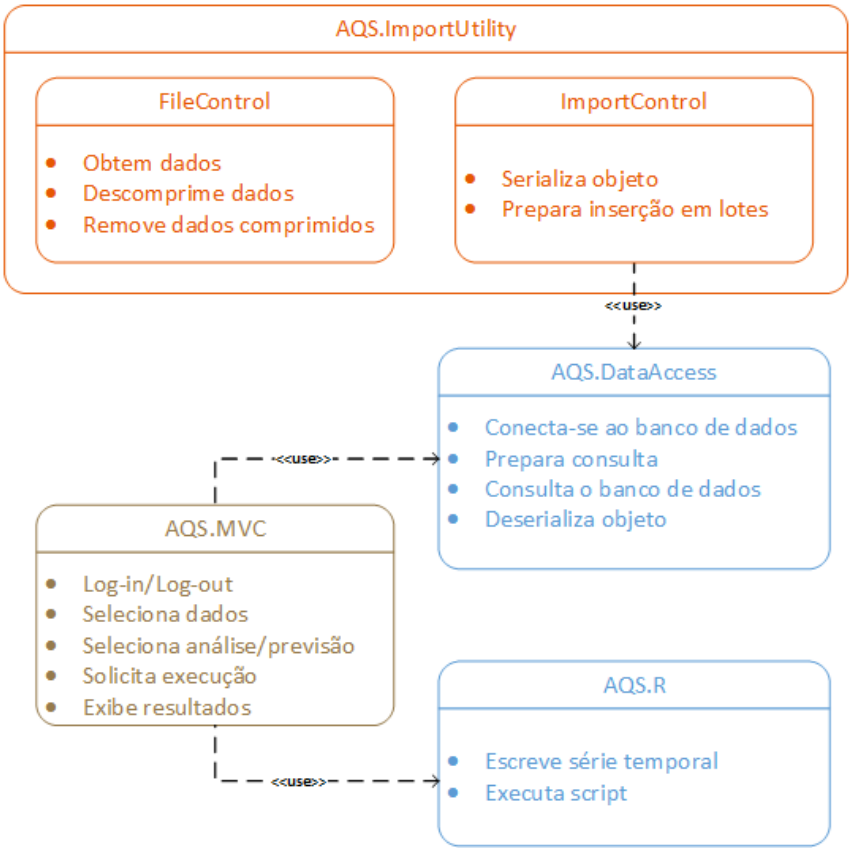

Figura 5: Diagrama de Módulos

Fonte: Autoria Própria

4 DESENVOLVIMENTO

Após a definição de como o sistema funciona quanto a sua estrutura, comportamento e interações, é necessário entender a sua estrutura real em termos de código e operações. Neste capítulo são apresentados os detalhes dos quatro módulos que integram o sistema.

4.1 AQS.IMPORTUTILITY

Este módulo tem como objetivo obter os dados disponíveis no website do AQS e preparar estes dados para que eles possam ser armazenados em instâncias locais, preparando-os para acessos futuros durante as etapas de análise e previsão. Nesta seção é explicado quais os tipos dos dados disponíveis, o seu conteúdo e como o sistema atua no seu processamento.

Os detalhes sobre os dados e os relacionamentos existentes entre eles, foram obtidos do documento AQS Data Dictionary (2011), o qual serviu como base para estruturar a forma como os dados são armazenados localmente.

4.1.1 TIPO DE DADOS

Os dados referentes às coletas de monitoramento do AQS estão distribuídos em diversas páginas de seu web site e disponíveis em dois tipos de arquivos diferentes: arquivos CSV (*Comma Separated Values* ou, em português, valores separados por vírgula), extensão ".csv", e arquivos de texto, extensão ".txt". Cada um desses arquivos contém dados que precisam ser processados de um modo diferente.

Os arquivos CSV contêm metadados que são importantes para melhorar a exibição da análise dos dados, como nomes de estados, condados, sites, unidades e outros associados a seus respectivos códigos. Estes dados são bastante desacoplados e, portanto, podem ser importados antes dos demais dados. Os seguintes dados são provenientes de arquivos CSV:

- *Agencies* (agências);
- *Urban Areas* (áreas urbanas);
- *Cities* (cidades);
- *Counties* (condados);
- *States* (estados);
- *Duration* (durações);
- *Collection Frequencies* (frequências de coleta);
- *Parameters* (parâmetros);
- *Qualifiers* (qualificadores);
- *Units* (unidades).

O site do AQS contém um único arquivo com informação sobre todos os sítios e monitores de coletas distribuídos pelos Estados Unidos, Canadá, México e Porto Rico. Além deste arquivo, ele contém diversos arquivos com os dados de amostras coletadas para diversos poluentes de 1993 até 2013. Os dados de sítios e monitores estão altamente acoplados e por isso devem ser importados ao mesmo tempo. Os dados com os valores das amostras estão menos acoplados e por isso podem ser os últimos dados importados. Estes dados compõem a porção *Big Data* do sistema e serão importados para o banco de dados NoSQL MongoDB. Os dados para as seguintes entidades são provenientes de arquivos texto:

- *Blank data* (dados de coleta em branco);
- *Composite data* (dados de coleta compostos);
- *Georeferences* (georeferências);
- *Locations* (localizações);
- *Roles* (papéis das agências de monitoramento);
- *Monitor collocation period* (período de disposição de monitores de coleta);
- *Monitor objectives* (objetivo dos monitores de coleta);
- *Monitor obstructions* (obstrução de monitores de coleta);

- *Monitor pollutant area* (áreas de monitores de coleta por poluentes);
- *Monitor protocols* (protocolos de monitores de coleta);
- *Monitor regulatory compliance* (conformidade regulamentar de monitores de coleta);
- *Monitors* (monitores de coleta);
- *Tangent roads* (estradas tangenciais a monitores de coleta);
- *Monitor types* (tipos de monitores de coleta);
- *Open paths* (caminhos abertos);
- *Raw data* (dados de coleta brutos);
- *Sample periods* (períodos de amostras);
- *Sample schedule* (planejamento de coletas);
- *Sites* (sítios);
- *Tangent Roads* (estradas tangenciais).

4.1.2 OBTENDO OS DADOS

O tamanho dos arquivos disponíveis no site do AQS para *download* variam muito. Alguns arquivos apresentam poucos kilobytes de dados, enquanto outros têm aproximadamente meio gigabyte de conteúdo. Para facilitar o processo de obtenção de dados, um *web crawler* - *software* para auxiliar a obtenção de dados de fontes *online* - foi desenvolvido para executar esta tarefa.

Um *web crawler* é um *software* desenvolvido para navegar em sites a procura de informações. Em geral, são utilizados para indexar a web, trabalhando por trás das ferramentas de busca. Muitos estudos são realizados para a melhoria e o desenvolvimento dos *web crawlers*, como nos trabalho de Castillo (2005) e Cho et al. (2001).

Para obter os dados, o *web crawler* deverá acessar a página de código e descrições do AQS (AQS Codes and Descriptions, 2014) para arquivos CSV e a página de dados detalhados do AQS para arquivos de texto (Download Detail AQS Data, 2014).

4.1.3 IMPORTANDO OS DADOS

Para importar os dados, três processos foram desenvolvidos. O primeiro importa os arquivos CSV, o segundo importa os arquivos de texto que contém dados sobre sítios e monitores e o terceiro importa os dados das amostras para o MongoDB.

Importar os dados CSV é um processo direto. A aplicação lê os arquivos, analisa cada linha, criando um registro se o registro ainda não existe, e armazena este registro.

Já para importar os arquivos de sítios e monitores, os dados que estavam presentes em um único arquivo são divididos em vários arquivos, agrupando-os pela entidade que cada um representa. Após este processo, são lidos, analisados e armazenados.

Por fim, para importar os dados das amostras e exportá-las para o MongoDB, cada arquivo é lido e os dados são agrupados por sítio, condado, estado, unidade e método de coleta. Estes grupos apresentaram um campo para os valores da amostra. Os valores são uma dupla chave-valor, no qual a chave é a data de coleta e o valor é o valor da amostra do poluente em questão.

Um registro, portanto, terá o formato conforme apresentado na Figura 6:

```
"_id" : ObjectId,
"StateCode" : "06",
"CountyCode" : "067",
"SiteID" : "0010",
"Parameter" : "42101",
"POC" : "1",
"SampleDuration" : "1",
"Unit" : "007",
"Method" : "011",
"Sample" : [
  {
    "Date" : "19930101",
    "StartTime" : "00:00",
    "SampleValue" : "0.0",
    "NullDataCode" : "",
    "SamplingFrequency" : "",
    "MonitorProtocolId" : "",
    "Qualifier1" : "",
    "Qualifier2" : "",
    "Qualifier3" : "",
    "Qualifier4" : "",
    "Qualifier5" : "",
    "Qualifier6" : "",
    "Qualifier7" : "",
    "Qualifier8" : "",
    "Qualifier9" : "",
    "Qualifier10" : "",
    "AlternateMethodDetectableLimit" : "",
    "Uncertainty" : "",
  },
  {
      "Date" : "19930101",
    "StartTime" : "01:00",
      ...
  }
  ...
        ]
```

Figura 6: Registro no MongoDB

Fonte: Autoria Própria

4.1.4 ADMINISTRANDO *BIG DATA* ATRAVÉS DO MONGODB

A parte do sistema que o caracteriza como um sistema *Big Data* e, portanto, requer que tecnologias como MapReduce e NoSQL sejam utilizadas para aumentar a eficiência, executa

suas operações de dados através um banco de dados MongoDB (MongoDB, 2014).

Este banco de dados utiliza técnicas de armazenamento orientado a documentos, os quais são formatados como JSONs binários (Binary JSON, 2014), e nos quais os atributos podem ser indexados livremente. O MongoDB ainda conta com suporte para replicação, fragmentação, disponibilização e consulta.

Este banco de dados, por ser não relacional e, consequentemente, ter seus dados organizados de forma não estruturada, não precisa executar operações de bancos de dados tradicionais, como operações de união e interseção.

4.2 AQS.MONGOACCESS

O módulo AQS.MongoAccess foi desenvolvido para se comportar com uma camada de acesso entre o banco de dados MongoDB e a aplicação desenvolvida em C#, a qual controla o fluxo dos dados entre os processos. Apesar de ser uma biblioteca de classes .NET bem simples, este módulo é muito importante para o sistema como um todo. Esta biblioteca de classes utiliza o driver oficial dos desenvolvedores do MongoDB para C# / .NET (C# and .NET MongoDB Driver, 2014). O módulo possui três classes, as quais são descritas e explicadas nas próximas seções:

- MongoConnector.cs;
- MongoReader.cs;
- MongoTimeSeries.cs.

Com a finalidade de melhorar a performance das consultas, os dados são indexados. O MongoDB utiliza índices como pequenas porções do conjuntos de dados, assim como índices em sistemas tradicionais de banco de dados, limitando o número de documentos que são inspecionados.

Respeitando a estrutura dos dados descrita na seção “Importando os dados”, três índices foram criados. O primeiro utiliza os campos “*StateCode*”e “*CountyCode*”, o segundo utiliza os mesmo campo e o campo “*SiteID*” e, por fim, o terceiro índice utiliza todos os campos, ou seja, “*StateCode*”, “*CountyCode*”, “*SiteID*”, “*Parameter*”, “*POC*”, “*SampleDuration*”, “*Unit*”, “*Method*”, “*Sample*”.

Os dois primeiros índices são usados na seleção parcial da séries temporais, quando o usuário seleciona hierarquicamente os campos, começando com a seleção do estado, em seguida

com a seleção do condado pertencente aquele estado e, por último, com a seleção do sítio disponível dentro daquele condado. Já o terceiro índice é utilizado para selecionar toda a série temporal.

4.2.1 *MONGOCONNECTOR*

A classe MongoConnector.cs é utilizada para conectar a aplicação em C# ao MongoDB. Ela contém uma instância do MongoClient, uma do MongoServer e uma do MongoDatabase. Estas classes estão disponíveis no driver. A conexão é estabelecida pelo MongoClient. A versão de desenvolvimento executa o MongoDB localmente, em mongodb://localhost. O acesso aos dados ocorre através do MongoDatabase.

Apesar da instância do MongoClient ser segura para ser utilizada em *threads* paralelas, o que permitiria que ela fosse utilizada como uma variável global, a implementação atual utiliza o padrão de projeto *singleton*. Esta escolha previne que um único usuário abra múltiplas conexões com o banco de dados. No entanto, cada usuário irá utilizar sua própria instância.

4.2.2 *MONGOREADER*

A classe MongoReader.cs é utilizada para recuperar os dados do banco de dados. Sua implementação atual consulta apenas a coleção *RawData*, a qual contém os dados brutos das observações. Quando o usuário cria uma instância do MongoReader, seu construtor chama o MongoConnector. O banco de dados possui outras duas coleções para dados: *BlankData* e *CompositeData*, e uma coleção para os dados dos usuários: *Users*. A demais coleções para dados não são utilizadas nesse momento porque os dados obtidos para fins de desenvolvimento e testes são todos do tipo *RawData*.

Uma vez que uma instância do MongoReader está disponível, é possível definir os parâmetros para a consulta através deste objeto. Se o usuário deseja, por exemplo, consultar o estado do Alabama e o condado de Maricopa, ele poderá utilizar o código como demonstrado na Figura 7:

```
AQSMongoReader mongoReader = new AQSMongoReader();

mongoReader.StateCode = "04";
mongoReader.CountyCode = "013";

List<MongoTimeSeries> ts = mongoReader.QueryRawData();
```

Figura 7: Código de execução do *AQSMongoReader*
Fonte: Autoria Própria

Além dos campos referentes ao código do estado e ao código do condado, o usuário poderá filtrar sua consulta baseado nos seguintes atributos:

- Código do estado;
- Código do condado;
- Código do sítio;
- Poluente;
- Código de ocorrência do poluente;
- Duração da amostra;
- Unidade;
- Método;
- Data de inicio da amostra;
- Data de fim da amostra.

Quando o usuário escolhe uma data de início e uma data de fim, toda a amostra é consultada e serializada como um objeto da aplicação. Isto é necessário porque a versão atual do MongoDB não permite a criação de *views* para atributos internos. A necessidade para esta funcionalidade já foi reportada e estará disponível em versões futuras do cliente (Support for selecting array elements in return specifier, 2014). Isto posto, as datas são filtradas na aplicação, após a leitura do banco de dados.

4.2.3 *MONGOTIMESERIES*

O resultado da consulta do MongoReader é uma lista de objetos da classes MongoTimeSeries. Estes objetos possuem as propriedades listadas na seção anterior e os valores da amostras, os quais possuem data, hora e o valor monitorado.

4.2.4 ACESSO AOS DADOS NO CONTEXTO DA *BIG DATA*

Os esforços aplicados para compreender os dados que estão disponíveis e exportá-los para uma base de dados NoSQL local foram fundamentais para estruturar a base do sistema e as propriedades *Big Data* relacionadas a ele, como armazenamento, estruturação e recuperação dos dados. O módulo de importação poderá ser executado para replicar a base de dados em outras máquinas. Já o módulo de acesso aos dados pode ser utilizado por outros programas que precisarem acessá-los. Neste sistema, o núcleo da aplicação web irá utilizá-lo para acessar os dados e enviá-los para o módulo com a capacidade de executar funções em R para processá-los.

4.3 AQS.R

O módulo AQS.R foi concebido como um componente para executar *scripts* da linguagem de programação estatística R através da aplicação em C#. O seu desenvolvimento e as melhorias aplicadas mostraram que este pacote pode ser utilizado como uma biblioteca de classes para outros projetos além deste sistema. Assim, os nomes, localizações e algumas definições em códigos são temporárias.

Os núcleos desses módulos são as classes ScriptExecutos.cs e Control.cs, as quais recebem argumentos para executar *scripts* em R e definem os argumentos da execução, respectivamente. Outras classes são utilizadas para configurar os parâmetros dos *scripts*.

4.3.1 ORGANIZAÇÃO DO MÓDULO

No diretório raiz do módulo estão as duas classes principais, como mencionado acima. Além destas, há outros seis pacotes: *Data*, *Utils*, *Openair*, *VARModeling*, *Forecast* e *Verification*. Os últimos quatro contém classes que herdam da classe *Control*, implementando um container para um *script* R específico, o qual permite que cada uma destas classes chamem o método *ExecuteScript* na classe ScriptExecutor.cs. O diagrama da Figura 8 mostra estes componentes e como eles se relacionam no diretórios raiz.

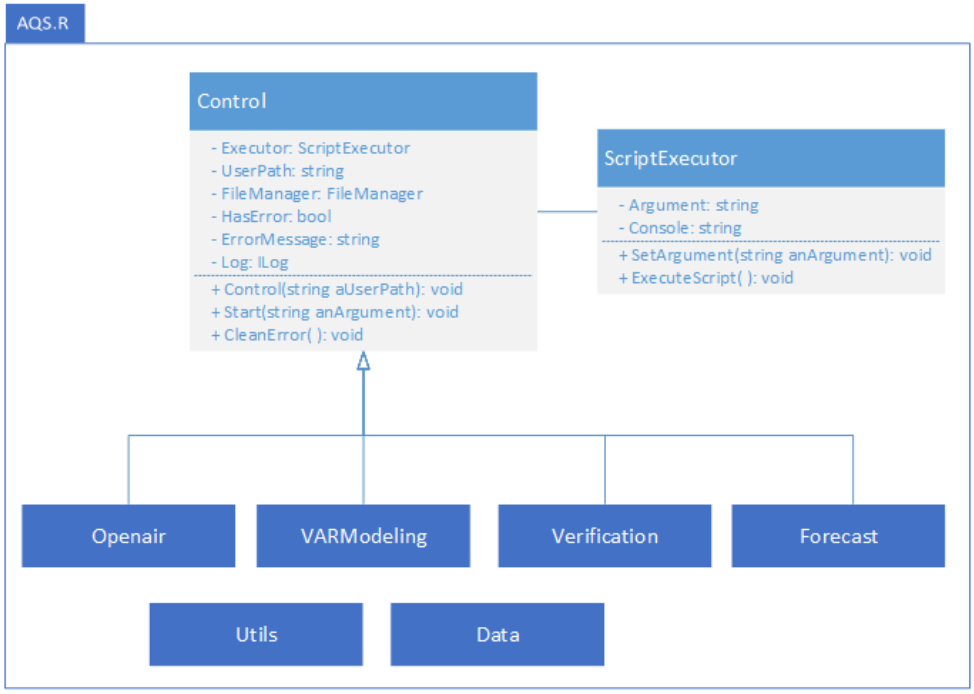

Figura 8: AQS.R

Fonte: Autoria Própria

4.3.2 *CONTROL*

Afim de organizar os scripts que serão executados, a classe Control.cs foi criada. Esta classe contém as informações básicas que as classes que implementam um *script* devem ter. Ela contém uma instância do ScriptExecutor, uma string com o diretório do usuário, para que este possa utilizar este diretório para armazenar dados sem comprometer os dados dos demais usuários, e uma variável que indica se há erro ou não, bem como uma variável com a mensagem de erro, se necessário. Os métodos que estão disponíveis nestas classes são comuns para todas as classes que executam *scripts*. São eles:

- *Start*: inicia a execução de um *script*;
- *ClearTemporaryData*: limpa todos os dados utilizados na execução de um *script*;
- *ClearError*: limpa os erros para que essa instância possa ser utilizada novamente.

4.3.3 *SCRIPTEXECUTOR*

Apesar da linguagem R ser mais utilizada como um ambiente de programação para calcular estatísticas, ela permite a execução de *scripts*. Para executar um *script* através do

programa, é necessário iniciar um processo como se um usuário estivesse executando o *script* no *prompt* de comando, por exemplo:

>> **Rscript MyCustomScript.r [arguments]**

Para ser capaz de executar tal ação em C#, um objeto *Process*, das bibliotecas padrões disponíveis na plataforma .NET, é criado e suas variáveis são definidas com as propriedades dos *scripts*. Na classe ScriptExecutor.cs, isto é feito da forma apresentada na Figura 9:

```
1    Process process = new Process();
2    process.StartInfo.FileName = Links.Rscript;
3    process.StartInfo.Arguments = Argument;
4    process.StartInfo.CreateNoWindow = true;
5    process.StartInfo.UseShellExecute = false;
6    process.StartInfo.RedirectStandardOutput = true;
7    process.Start();
8
9    Console = process.StandardOutput.ReadToEnd();
10   process.WaitForExit();
```

Figura 9: Executando um *script* através de um *Process*

Fonte: Autoria Própria

Este processo executará o *script* R indicado na variável *Argument*. Se, após a execução, for necessário verificar a saída do console, o usuário poderá verificar a variável nomeada Console.

4.3.4 PACOTE *DATA*

O pacote *Data* tem duas classes: TimeSeries.cs, a qual define a estrutura de dados que são utilizado em todos os processos do AQS.R, e TimeSeriesControl.cs, a qual contém métodos para ajustar duas ou mais séries temporais. Estes métodos ajustam a freqüência, o tamanho da amostra, as unidades e os nomes dos poluentes. Essas classes estão presentes no pacote como apresentado na figura 10:

Figura 10: AQS.R - Data

Fonte: Autoria Própria

Para criar um série temporal, o usuário poderá acessar três construtores diferentes.

O primeiro cria uma série temporal vazia, como apresentado na Figura 11.

```
1 TimeSeries timeSeries = new TimeSeries("\textit{Carbon Monoxide}");
```

Figura 11: Criando uma série temporal vazia

Fonte: Autoria Própria

O segundo, uma série temporal baseada em um objeto da classe MongoTimeSeries, como na Figura 12.

```
AQSMongoReader mongoReader = new AQSMongoReader();

mongoReader.StateCode = "04";
mongoReader.CountyCode = "013";
mongoReader.SiteID = "0019";
mongoReader.Method = "011";

mongoReader.BeginDate = new DateTime(1993, 1, 1);
mongoReader.EndDate = new DateTime(1993, 6, 30);

List<MongoTimeSeries> tsList = mongoReader.QueryRawData();
TimeSeries timeSeries = new TimeSeries(tsList[2]);
```

Figura 12: Criando uma série temporal com *MongoTimeSeries*

Fonte: Autoria Própria

E o último cria um série temporal a partir de uma lista de valores de amostra, como na Figura 13.

```
List<double> values = new List<double> { 0.1, 1.4, 0.5, 6.8 };
DateTime start = new DateTime(2014, 05, 1);
TimeSeries ts = new TimeSeries(start, Frequency.Hourly, "008", values
    );
```

Figura 13: Criando uma série temporal com valores

Fonte: Autoria Própria

A classe TimeSeriesControl.cs geralmente é utilizada dentro dos métodos de classes que herdam da classes Control.cs para ajustar e escrever valores da série temporal para um arquivo, o qual será lido por um *script* quando este for ser executado. Esta utilização é apresentada na Figura 14.

```
TimeSeriesControl.PrepareTimeSeries(timeSeries);
TimeSeriesControl.WriteTimeSeries(UserPath, timeSeries)
```

Figura 14: Usando *TimeSeriesControl*

Fonte: Autoria Própria

4.3.5 PACOTE *UTILS*

O pacote Utils oferece várias ferramentas pra tornar o uso do código mais fácil. Ele contém métodos que podem ser utilizados por outras classes do módulo AQS.R. As classes do Utils são apresentadas no diagrama da figura 15. Este pacote contém uma classe para criar *logs* de execução, uma classe para armazenar links externos, uma classes para centralizar enumeradores, uma classe para guardar constantes, uma classe para converter dados entre diferentes formatos, uma classe para gerenciar unidades, e uma classe para gerenciar arquivos externos.

Figura 15: AQS.R - Utils

Fonte: Autoria Própria

4.3.6 PACOTE *FORECAST*

O pacote *Forecast* foi criado para organizar as classes que empacotam diferentes métodos do pacote *forecast, Forecasting functions for time series and linear models* (2014) para a linguagem R. Estes métodos são implementados em *scripts*. Cada classes irá prover os parâmetros necessários para a execução deles. O pacote pode ser visualizado na figura 16:

Figura 16: AQS.R - Forecast

Fonte: Autoria Própria

4.3.6.1 *FORECAST*

A classe *Forecast* permite que o usuário realize a previsão de dados para séries temporais univariadas. O usuário pode ajustar o atributo *Ahead* conforme o número de períodos de tempo a frente que ele deseja prever. O usuário também pode ajustar o método que deseja utilizar. O pacote *forecast* oferece os métodos: *ets*, o qual realiza previsão através de um amortecimento exponencial, no qual os parâmetros que indicam se o armotecimento é aditivo ou multiplicativo são configurados automaticamente; *arima*, o qual calcula a previsão através de uma média móvel autoregressiva; *naive*, o qual é um container para a previsão *arima*, oferecendo ajustes para uma previsão *naive*; e *rwdrift*, o qual também é um container para *arima*, oferecendo ajustes para uma previsão com *random walk* (HYNDMAN, 2014).

Após a chamada do método *Execute*, o resultado será representado em um gráfico e os valores numéricos estarão disponível em uma lista de *doubles* na variável *ForecastResult*. O código pode ser observado na Figura 17.

```
Forecast forecast = new Forecast(userPath);
forecast.SetTimeSeries(new TimeSeries(ts[0]));
forecast.Ahead = new TimeSpan(48, 0, 0);
forecast.Method = ForecastMethods.ETS;
forecast.Execute();
```

Figura 17: Código para execução do método *Forecast*

Fonte: Autoria Própria

O resultado da execução irá criar o gráfico como o demonstrado na Figura 18. Este gráfico mostra o último dia da série temporal monitorada escolhida pelo usuário e a previsão para os n dias seguintes, sendo n o número de dias escolhidos para a previsão. O resultado é indicado dentro de dois intervalos de confiança, o primeiro de 80% e segundo de 95%.

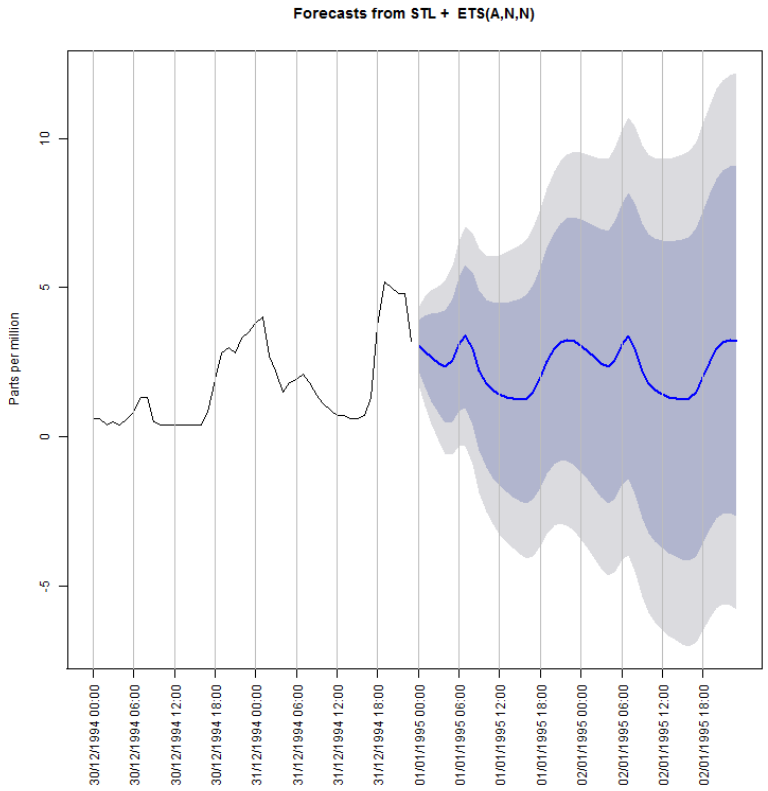

Figura 18: AQS.R - Forecast

Fonte: Autoria Própria

4.3.7 PACOTE *VARMODELING*

O pacote *VARModeling* pode ser utilizado para prever dados utilizando séries temporais multivariadas. As classes neste pacote são utilizadas como containers para métodos do pacote *vars: VAR Modelling* (2014) da linguagem R.

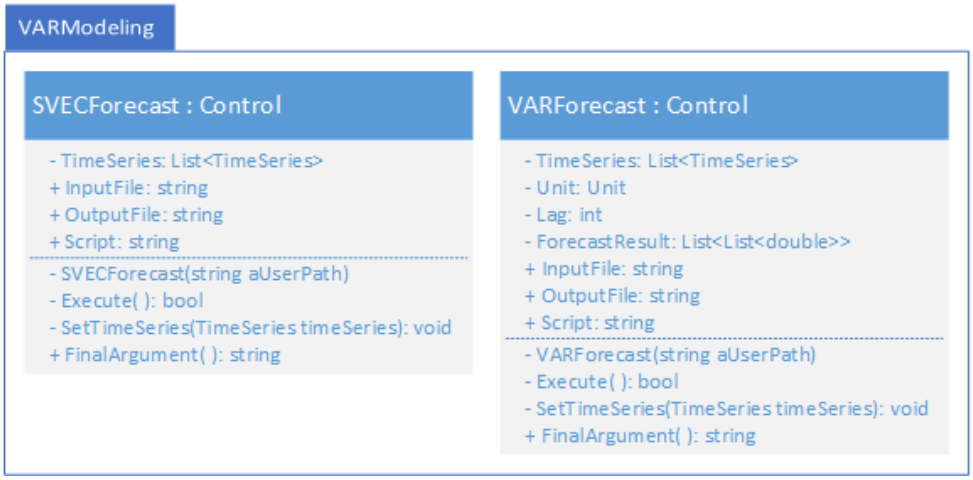

Figura 19: AQS.R - VARModeling

Fonte: Autoria Própria

4.3.7.1 *SVECFORECAST*

A classe *SVECForecast* estima um modelo estrutural de correção de erros, do inglês *Structural Vector Error Correction Model* (SVEC), para duas séries temporais.

Este modelo calcula a cointegração entre as duas séries temporais, revelando se as séries possuem um relacionamento a longo prazo. O resultado será a resposta de uma variável em relação a outra. A Figura 21 mostra um exemplo do resultado esperado para o trecho de código da Figura 20.

```
1 SVECForecast svec = new SVECForecast(userPath);
2 svec.SetTimeSeries(t1, t2);
3 svec.Execute();
```

Figura 20: Código para execução do método *SVEC*

Fonte: Autoria Própria

Na Figura 21, é possível observar que há uma resposta positiva da primeira série temporal de monóxido de carbono (COA - *Carbon Monoxide A*) em relação à segunda (COB - *Carbon Monoxide B*). Isto indica que o aumento de uma série está relacionado com o aumento da outra.

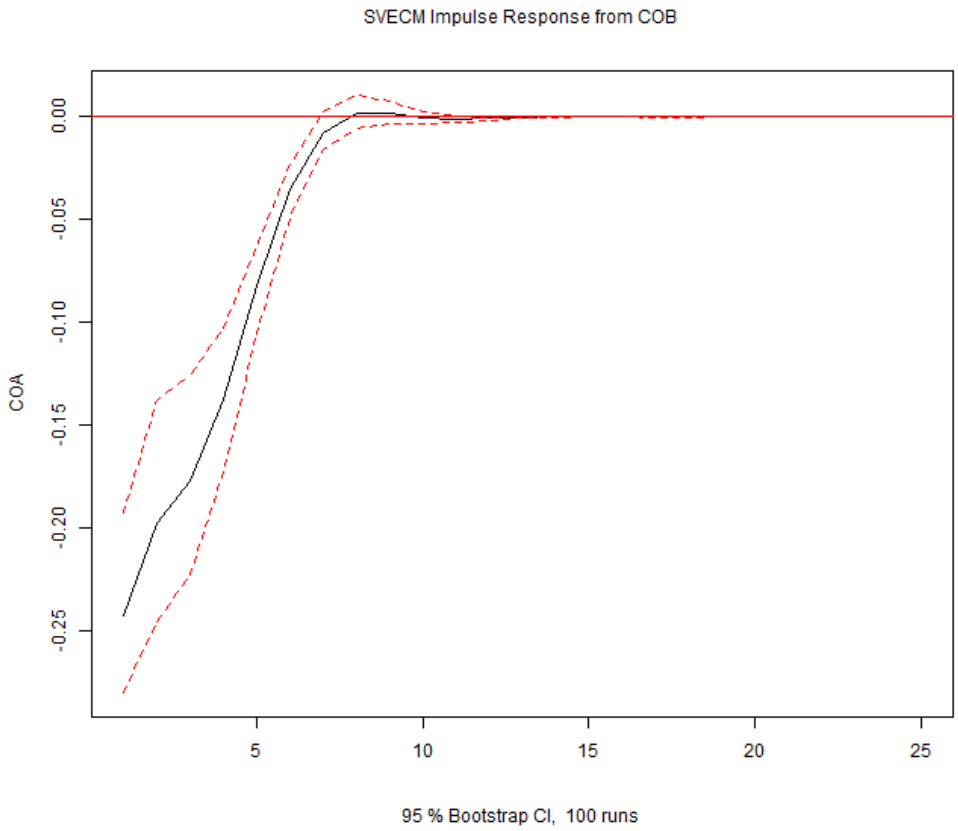

Figura 21: AQS.R - SVECForecast
Fonte: Autoria Própria

4.3.7.2 *VARFORECAST*

Esta classe implementa o modelo VAR, do inglês *Vector Autoregressive Model* (VAR). Para utilizá-lo, o usuário deve escolher as séries temporais e o *lag*, sendo que a primeira série temporal é prevista e as demais são utilizadas para a criação do modelo.

O *lag* diz ao *script* quantas unidades de tempo do passado devem ser consideras para criar o modelo. O usuário poderá chamar o método *VARSelect* para obter uma sugestão. Este método precisa de um valor máximo para realizar sua busca. O resultado é uma lista de quatro inteiros, cada um de acordo com um critério: *Akaike*, *Hannan-Quinn*, *Schwarz* e

Forecast Prediction Error. A chamada do método para executar o *script* não irá desenhar os resultados em um gráfico automaticamente. Os valores podem ser acessados através da variável ForecastResults. O trecho de código na Figura 22 é um exemplo de como utilizar o método *VARSelect*.

```
VARForecast varForecast = new VARForecast(userPath);
varForecast.SetTimeSeries(t1, t2);
List<int> result = varForecast.VARSelect(30);
varForecast.Lag = result.Max();
varForecast.Execute();
```

Figura 22: Código para execução do método *VARSelect*

Fonte: Autoria Própria

Os resultados podem ser utilizados para criar uma nova série temporal, a qual pode ser desenhada para compará-la com dados monitorados no mesmo período utilizando a classe Comparison.cs, a qual será explicada em seções seguintes.

4.3.8 PACOTE *OpenAir*

Este módulo implementa vários métodos do pacote OpenAir (2014) para a linguagem R. Estes métodos são muito importantes para analisar a qualidade de dados de poluição do ar. O diagrama da Figura 23 mostra as classes neste pacote.

Detalhes de cada métodos, bem como outros parâmetros que podem ser adicionados para versões futuras podem ser encontrados no *OpenAir Manual* (CARSLAW; ROPKINS, 2012), disponível para download no website do OpenAir. Esta seção mostra os métodos implementados pela versão atual do AQS.R.

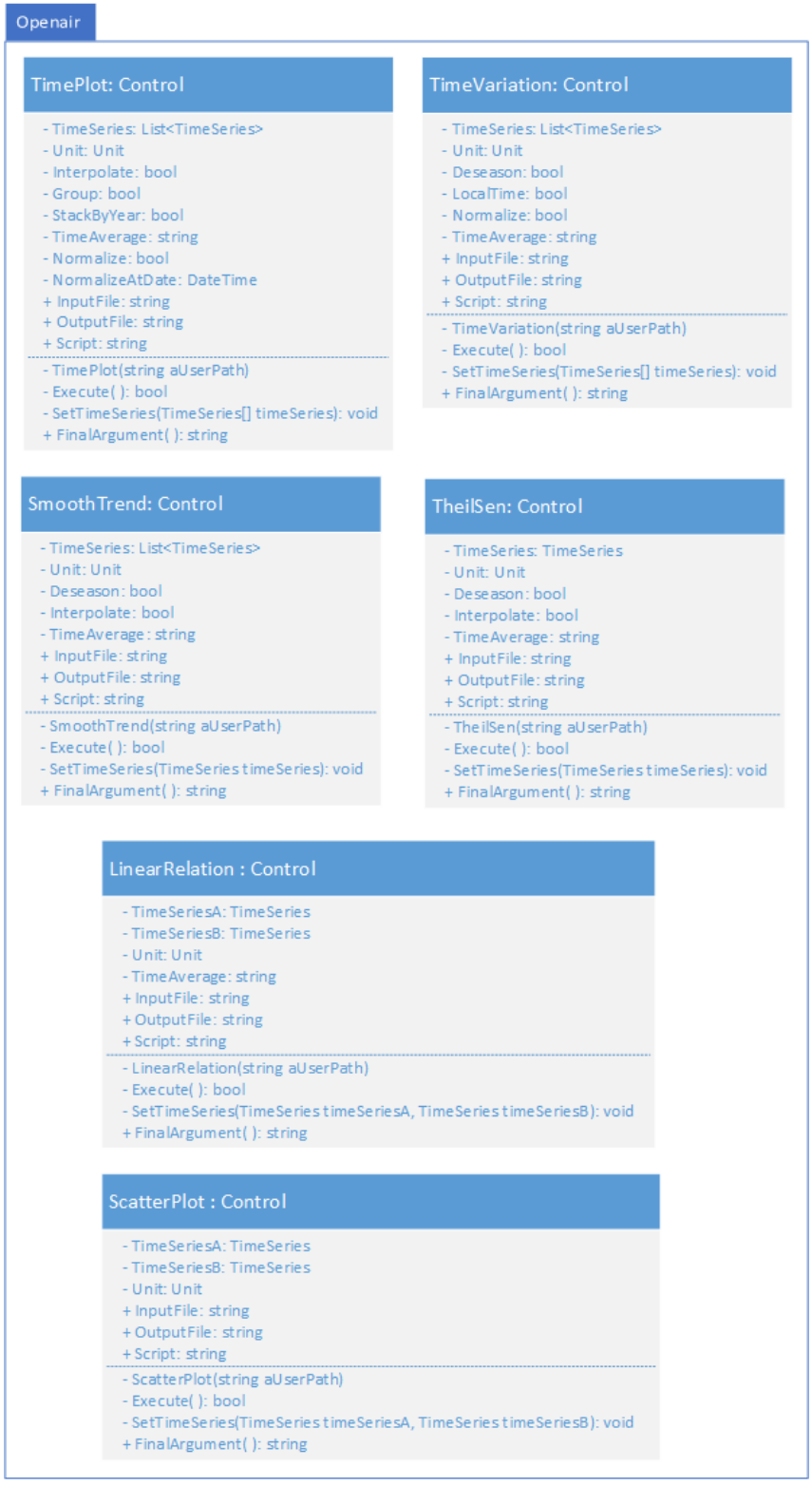

Figura 23: AQS.R - Openair

Fonte: Autoria Própria

4.3.8.1 *TIME PLOT*

Através do pacote *Openair* para a linguagem R é possível chamar a função *timePlot(...)*. Tendo em vista que esta função tem vários argumentos que modificam sua saída, a classes em C# TimePlot.cs, a qual herda da classe Control.cs, irá funcionar como um adaptador. Portanto, todos os parâmetros podem ser ajustados no código em C#, fazendo com que os *scripts* em R se comportem como esperado.

O usuário pode ajustar a variável *Grouped*, a qual define se os dados serão agrupados no mesmo gráfico, ou se cada poluente terá seu próprio gráfico. Já a variável *Stacked* define se os dados serão agrupados por ano. O usuário pode ainda ajustar a média que será utilizada para calcular o resultado.

Estas variáveis permitirão diferentes saídas para o mesmo método. Por exemplo, o trecho da Figura 24 irá gerar o gráfico da Figura 25.

```
AQSMongoReader mongoReader = new AQSMongoReader();

mongoReader.StateCode = "04";
mongoReader.CountyCode = "013";

mongoReader.BeginDate = new DateTime(1995, 1, 1);
mongoReader.EndDate = new DateTime(1997, 12, 31);

List<MongoTimeSeries> ts = mongoReader.QueryRawData();

TimePlot timePlot = new TimePlot(@"~\\AQS.R\\UserData\\User1\\");

TimeSeries t1 = new TimeSeries(ts[2]);
TimeSeries t2 = new TimeSeries(ts[12]);

timePlot.SetTimeSeries(t1, t2);
timePlot.Execute();
```

Figura 24: Código para execução do método *Time Plot*

Fonte: Autoria Própria

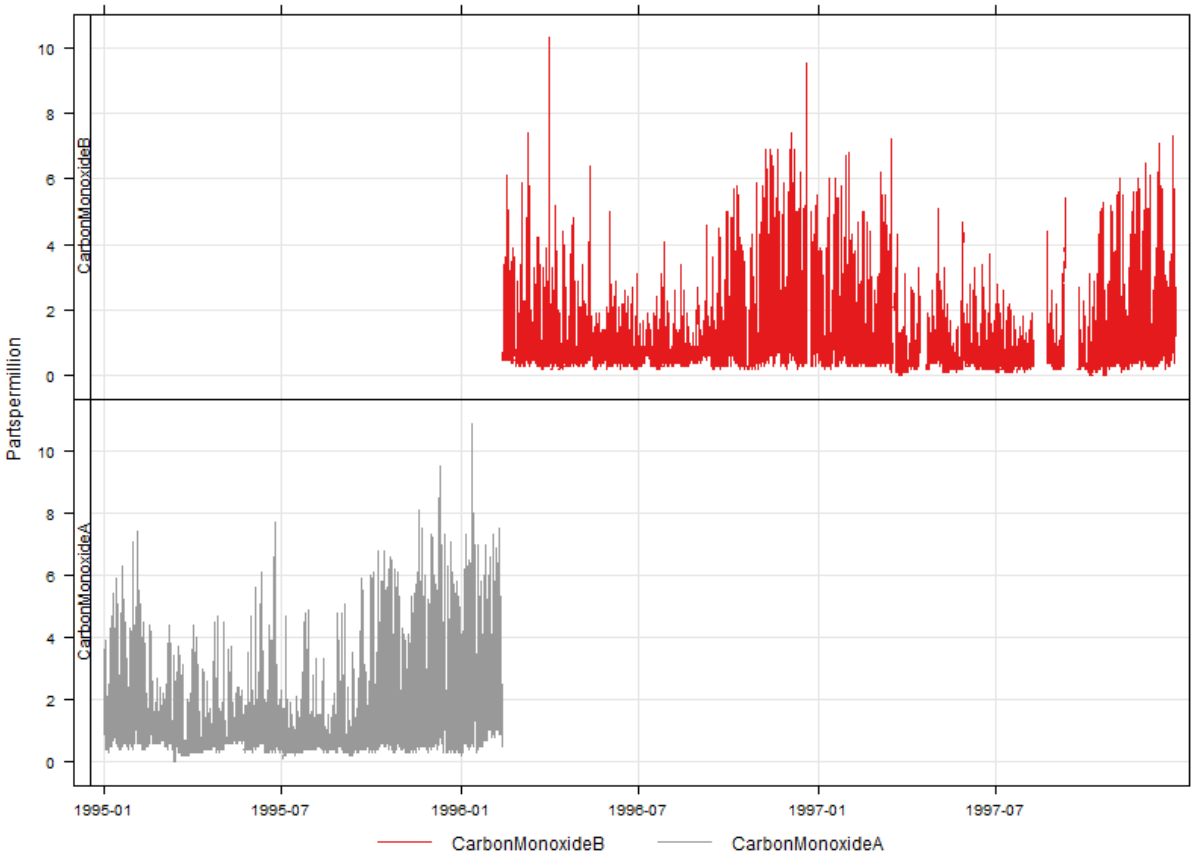

Figura 25: AQS.R - *Time Plot*

Fonte: Autoria Própria

O gráfico da Figura 25 apresenta duas séries temporais de monóxido de carbono, monitoradas por hora em partes por milhão (ppm) de primeiro de janeiro de 1995 a 31 de dezembro de 1997. Adicionando as linhas de código da Figura 26, o gráfico irá mudar para o gráfico da Figura 27.

```
timePlot.Group = true;
timePlot.TimeAverage = Average.Month;
```

Figura 26: Alteração no código para execução do método *Time Plot*

Fonte: Autoria Própria

No gráfico da Figura 27, as mesmas séries temporais são apresentadas, no entanto, elas foram agrupadas para compartilharem o mesmo eixo vertical e foram recalculadas para apresentarem a média mensal.

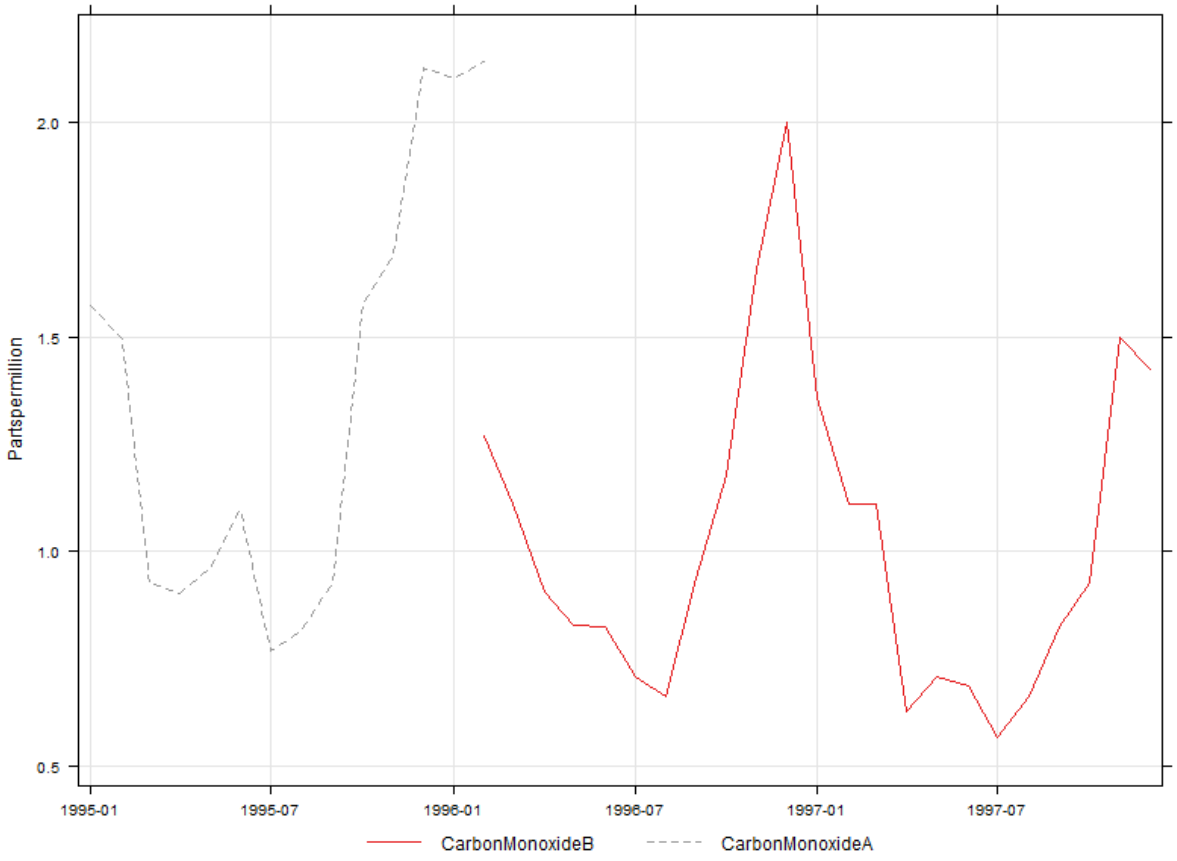

Figura 27: AQS.R - Time Plot Agrupado

Fonte: Autoria Própria

4.3.8.2 *TIME VARIATION*

A classe *TimeVariation* cria um gráfico que mostra o comportamento de várias séries temporais ao mesmo tempo. Os dados podem ser normalizados para serem desenhados no mesmo gráfico e darem sentido a comparações. O usuário também poderá ajustar o *script* para utilizar o horário local.

```
1 TimeVariation timeVariation = new TimeVariation(userPath);
2 timeVariation.SetTimeSeries(new TimeSeries(ts[2]), new TimeSeries(ts
     [3]));
3 timeVariation.LocalTime = true;
4 timeVariation.Normalize = true;
5 timeVariation.Execute();
```

Figura 28: Código para execução do método *Time Variation*

Fonte: Autoria Própria

A Figura 29 mostra o resultado do trecho de código da Figura 28. No resultado observa-se quatro gráficos diferentes. O gráfico superior apresenta a variação horária por dia da semana. O gráfico inferior esquerdo apresenta a variação horária total. Já o gráfico inferior central apresenta a variação mensal. Por fim, o gráfico inferior direito apresenta a variação semanal.

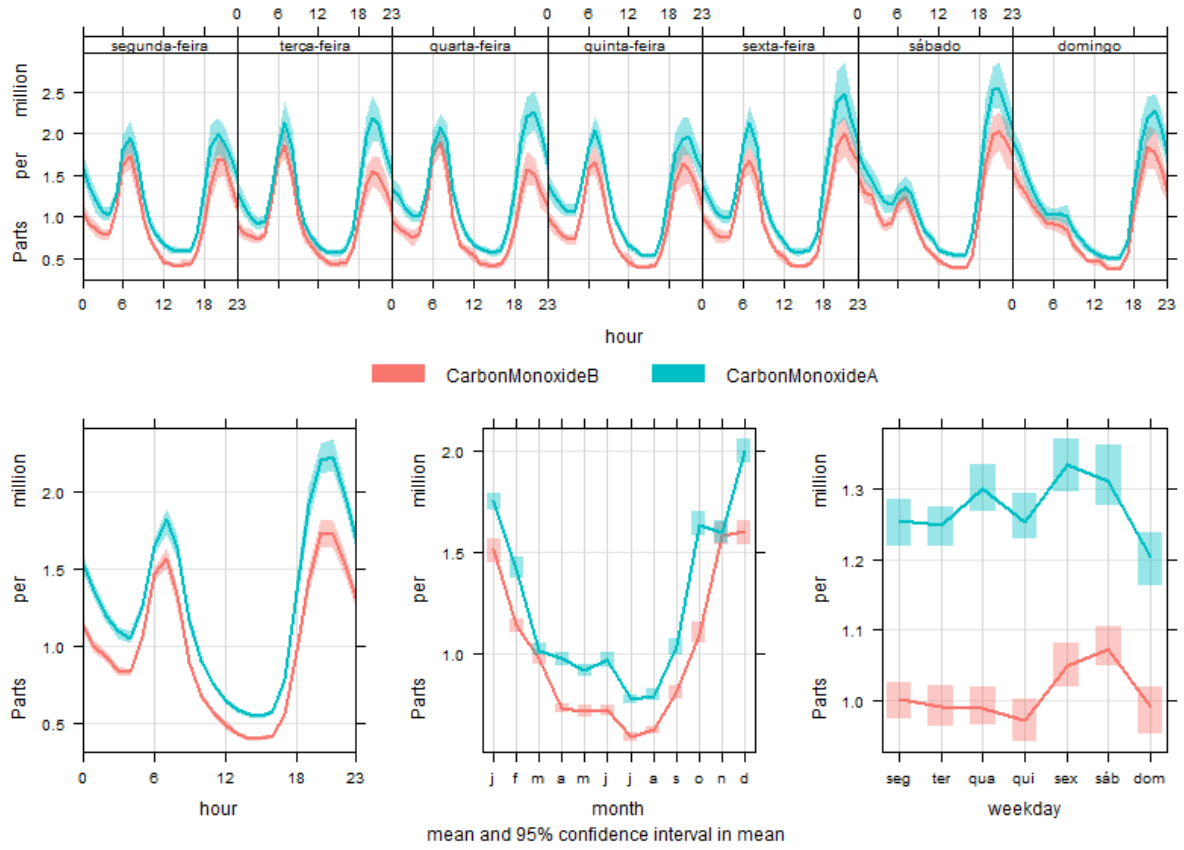

Figura 29: AQS.R - Time Variation

Fonte: Autoria Própria

4.3.8.3 *SMOOTH TREND*

Utilizando esta classe, o usuário pode calcular concentrações médias para as séries temporais. O usuário pode dar um valor para o atributo *TimeAverage* que pode ser *month*, *year* e *season*. O resultado mostra a curva e o intervalo de confiança de 95% de sua suavização.

```
SmoothTrend smoothTrend = new SmoothTrend(userPath);

TimeSeries t1 = new TimeSeries(ts[2]);
TimeSeries t2 = new TimeSeries(ts[12]);

smoothTrend.SetTimeSeries(t1, t2);
smoothTrend.Execute();
```

Figura 30: Código para execução do método *Smooth Trend*

Fonte: Autoria Própria

Utilizando-se médias mensais e dados com tendência de estação, observa-se o resultado da Figura 31 para o trecho de código da Figura 30.

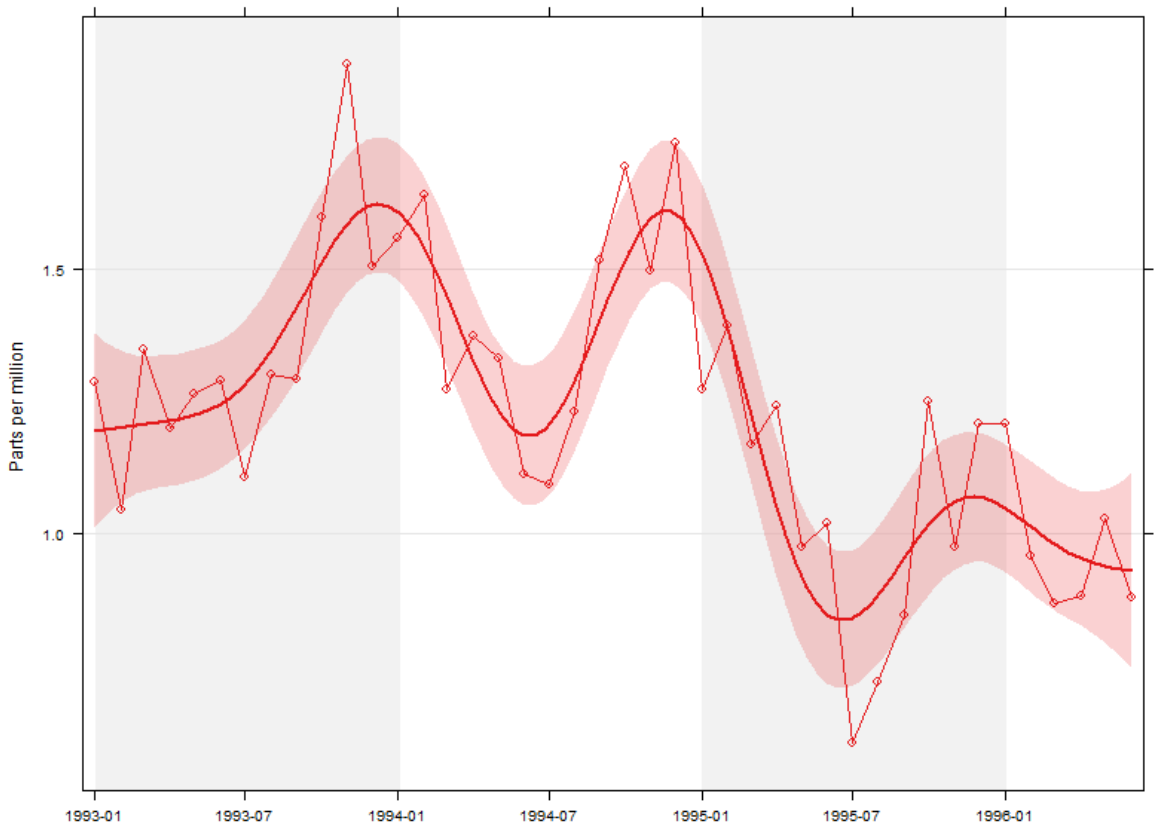

Figura 31: AQS.R - Smooth Trend

Fonte: Autoria Própria

4.3.8.4 *THEIL-SEN*

A classe *TheilSen* pode ser utilizada para chamar o método *TheilSen* no pacote *Openair*. Ele calcula picos entre todos os pares x e y fornecidos. O usuário pode escolher a média que quer, bem como se os componentes sazonais devem ser utilizados ou não. O seguinte código mostra um exemplo de como utilizar a classe TheilSen:

```
TheilSen theilSen = new TheilSen(userPath);
theilSen.SetTimeSeries(new TimeSeries(ts[2]));
theilSen.Deseason = false;
theilSen.Execute();
```

Figura 32: Código para execução do método *Theil-Sen*

Fonte: Autoria Própria

O código da Figura 32 irá produzir o resultado da Figura 33. O resultado apresenta a série temporal original e a sua tendência em um intervalo de confiança de 95%.

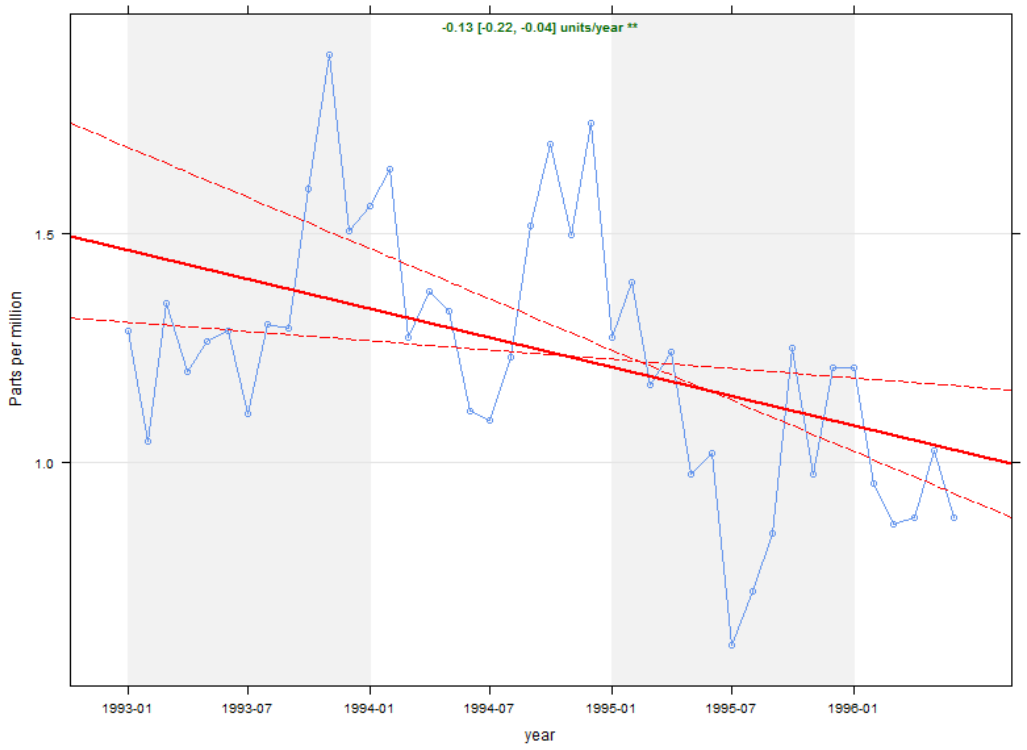

Figura 33: AQS.R - Theil-Sen

Fonte: Autoria Própria

4.3.8.5 *LINEAR RELATION*

A relação linear entre duas variáveis pode ser desenhada em um gráfico utilizando o método de mesmo nome do pacote *OpenAir*. Para fazê-lo, o usuário pode utilizar um código como o demonstrado na Figura 34:

```
LinearRelation linearRelation = new LinearRelation(userPath);
TimeSeries t1 = new TimeSeries(ts[2]);
TimeSeries t2 = new TimeSeries(ts[3]);
linearRelation.SetTimeSeries(t1, t2);
linearRelation.Execute();
```

Figura 34: Código para execução do método *Linear Relation*

Fonte: Autoria Própria

O resultado será o gráfico da Figura 35. Neste gráfico é apresentada a inclinação horária do relacionamento entre duas séries temporais, a primeira de monóxido de carbono e a segunda de dióxido de nitrogênio. Para cada período da inclinação horária calculado, é apresentado o intervaldo de confiança de 95%.

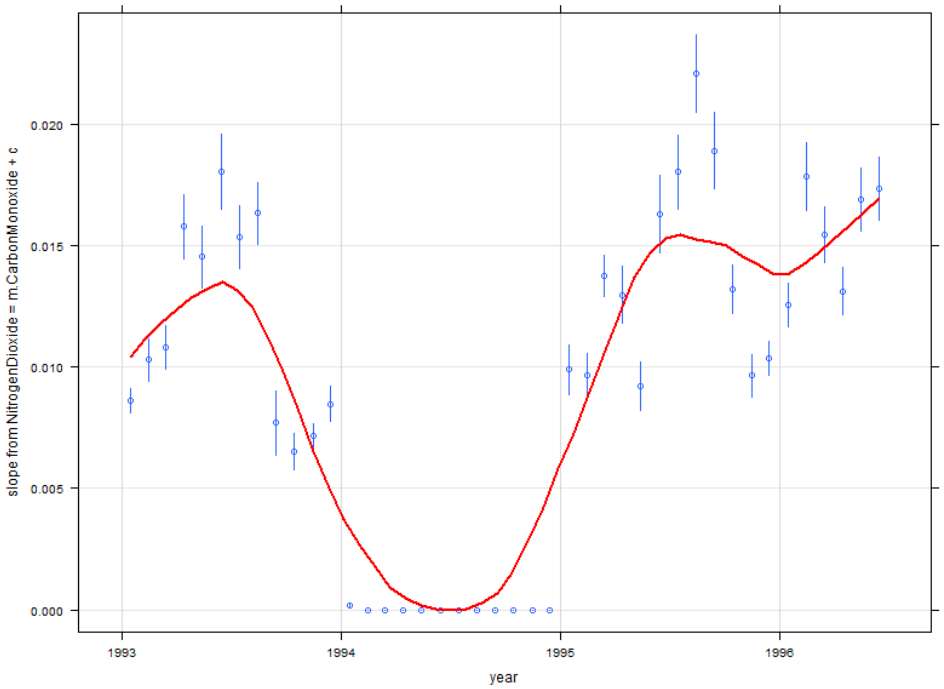

Figura 35: AQS.R - Linear Relation

Fonte: Autoria Própria

4.3.8.6 *SCATTER PLOT*

Finalmente, o pacote *Openair* pode ser utilizado para analisar a correlação entre duas variáveis através da análise *Scatter Plot*. O código para criação do gráfico da Figura 37 é apresentado na Figura 36.

```
ScatterPlot scatterPlot = new ScatterPlot(userPath);
TimeSeries t1 = new TimeSeries(ts[2]);
TimeSeries t2 = new TimeSeries(ts[3]);
scatterPlot.SetTimeSeries(t1, t2);
scatterPlot.Execute();
```

Figura 36: Código para execução do método *Scatter Plot*

Fonte: Autoria Própria

Um gráfico de dispersão demonstra o relacionamento entre duas variáveis. O resultado do exemplo, exibido na Figura 36, apresenta o relacionamento entre duas séries temporais de monóxido de carbono (COA e COB). Seguindo a escala de contagem apresentada ao lado direito do gráfico, indentifica-se uma relação maior e mais linear nos valores menores.

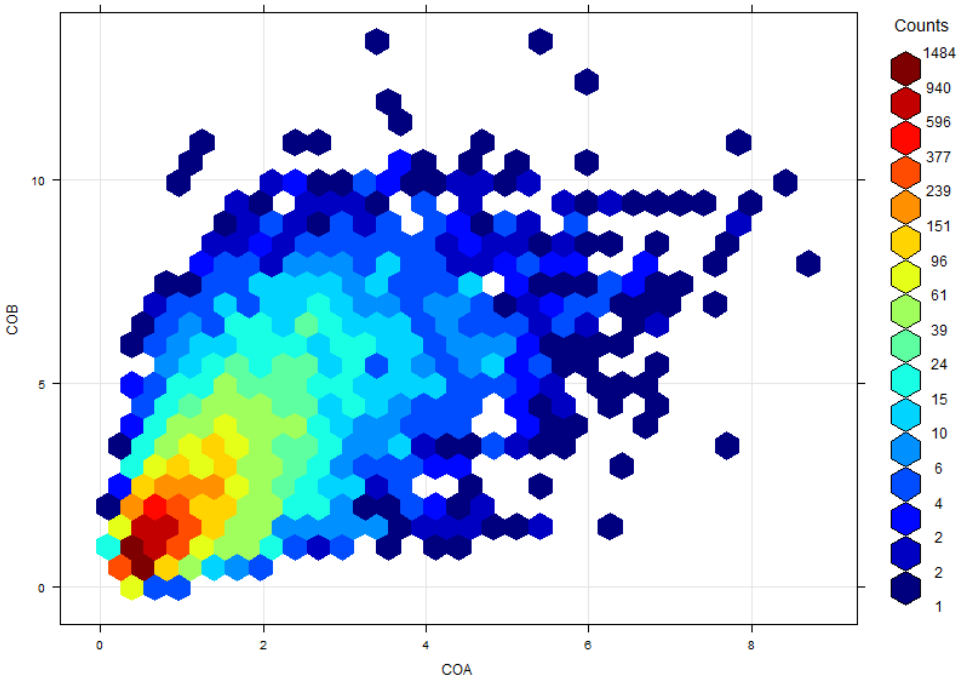

Figura 37: AQS.R - Scatter Plot

Fonte: Autoria Própria

4.3.8.7 *COMPARISON*

Depois de utilizar um dos métodos de previsão, o usuário poderá selecionar os dados reais monitorados para o dia da previsão e compará-los para que ele possa avaliar o quão bom é o novo modelo e o quanto ele pode confiar no modelo. A classe *Comparison* desenha ambos, a série temporal monitorada e a modelada, e calcula o erro quadrático médio, para que seja possível comparar diferentes modelos.

```
AQSMongoReader mongoReader = new AQSMongoReader();

mongoReader.StateCode = "04";
mongoReader.CountyCode = "013";
mongoReader.Method = "011";

mongoReader.BeginDate = new DateTime(1993, 1, 1);
mongoReader.EndDate = new DateTime(1993, 12, 31);

List<MongoTimeSeries> ts = mongoReader.QueryRawData();
TimeSeries t1 = new TimeSeries(ts[2]);
TimeSeries t2 = new TimeSeries(ts[3]);

VARForecast varForecast = new VARForecast(userPath);
varForecast.SetTimeSeries(t1, t2);
varForecast.Lag = varForecast.VARSelect(30).Max();
varForecast.Execute();

mongoReader = new AQSMongoReader();
mongoReader.StateCode = "04";
mongoReader.CountyCode = "013";
mongoReader.Method = "011";
mongoReader.BeginDate = new DateTime(1994, 1, 1);
mongoReader.EndDate = new DateTime(1994, 1, 1, 23, 0, 0);
List<MongoTimeSeries> tsCompare = mongoReader.QueryRawData();

TimeSeries measured = new TimeSeries(tsCompare[2]);

Comparison comparison = new Comparison(userPath);
comparison.SetTimeSeries(new TimeSeries(measured.Start(),Frequency.
   Hourly, measured.Unit.Code, varForecast.ForecastResult[0]),
   measured);
comparison.Message = "Forecast Method: VAR";
comparison.Execute();
```

Figura 38: Código para execução do método *Comparison*

Fonte: Autoria Própria

O código da Figura 38 compara os resultados de *VARForecast* com os dados reais.

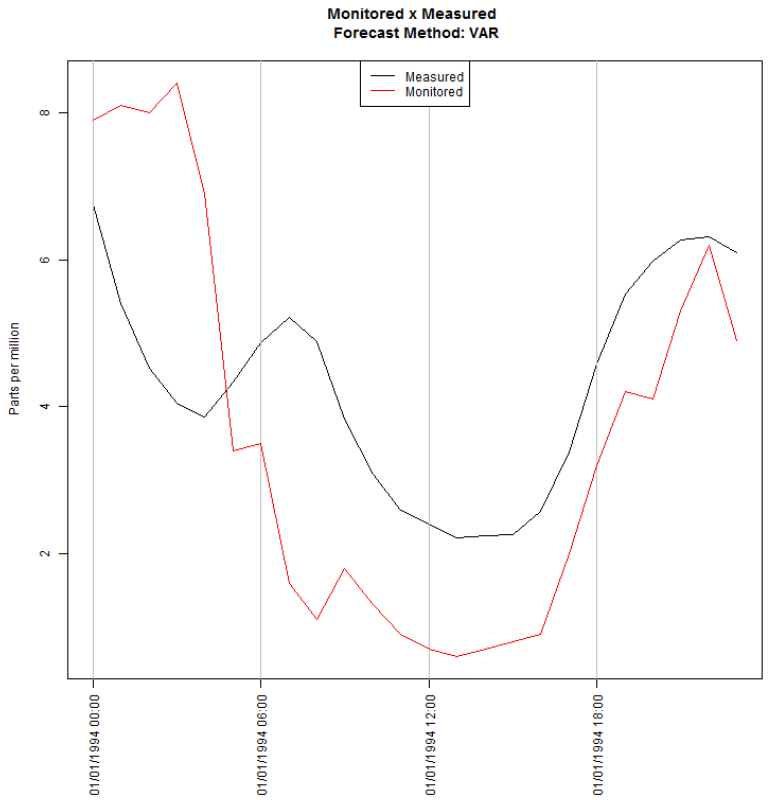

Figura 39: AQS.R - Comparison

Fonte: Autoria Própria

4.3.9 A LINGUAGEM R NO CONTEXTO DA *BIG DATA*

Apesar da necessidade da criação de várias estruturas para encapsular os métodos dos pacotes VAR, *OpenAir*, e *Forecast* afim de organizar o módulo AQS.R, este módulo apresenta uma estrutura bem simples e que da a oportunidade de criação de novas estruturas para novas análises utilizando *scripts* em R. A utilização da linguagem R é apropriada para sistemas *Big Data*, pois ela é estruturada para computar um grande volume de dados, além de oferecer vários pacotes com soluções prontas para diferentes tipos de dados. Com o acesso aos dados viabilizados através do módulo AQS.MongoAccess e o processamento, através do módulo AQS.R, torna-se necessário o desenvolvimento de um módulo que permita ao usuário utilizar o sistema de forma transparente.

4.4 AQS.MVC

Até este momento foram apresentados três módulos principais: AQS.ImportUtiliity, utilizado para importar os dados para o MongoDB, AQS.MongoAccess, utilizado para ler os dados e utilizá-los em memória e, por fim, AQS.R para executar as análises. Por fim, é necessário desenvolver o *front-end* do sistema, o qual permitirá que usuários acessem os dados e as análises. Nesta seção é apresentado o AQS.MVC, um módulo que oferece um meio para acesso.

Este módulo é uma interface web que emprega o método de desenvolvimento ASP.NET MVC4 (2014), utilizando o *framework* .NET 4.5. Utiliza também o Twitter Bootstrap (2014) e o MvcJqGrid (2014) para permitir uma aparência melhor do que HTML puro, enriquecendo a experiência do usuário. Neste momento a aplicação está disponível em um servidor web IIS local.

4.4.1 INICIANDO A APLICAÇÃO

A requisição da aplicação em um *browser* irá apresentar para o usuário a página para que seja realizado o acesso ao sistema. Esta página contém um formulário de acesso, para o qual o usuário deverá informar seu nome de usuário e sua senha para obter acesso, como demonstrado na Figura 40. Caso o usuário não esteja cadastrado, ele poderá se cadastrar na página de inscrição.

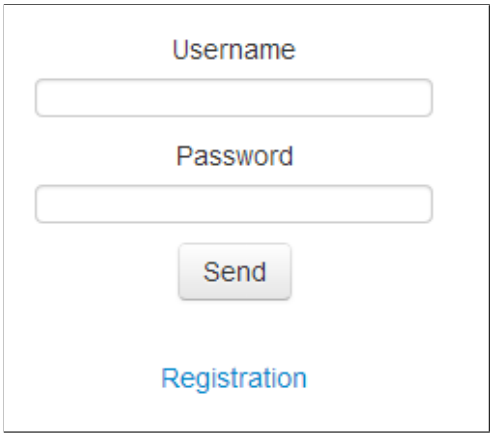

Figura 40: AQS.MVC - Login
Fonte: Autoria Própria

Para criar um novo registro, o usuário deverá informar um nome de usuário, o qual deverá ser único (o usuário será informado caso outro usuário esteja utilizando o nome

escolhido), o seu e-mail e uma senha. O formulário de registro pode ser visto na figura 41

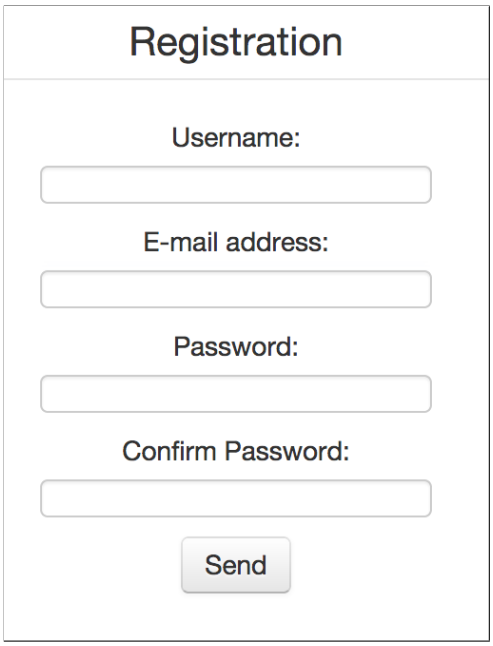

Figura 41: AQS.MVC - Registration
Fonte: Autoria Própria

Ao realizar uma nova inscrição ou utilizar credenciais válidas para acessar o sistema, o usuário é direcionado para a página principal da aplicação. Esta página possui três áreas: um menu fixo na parte esquerda, um painel de abas no centro e um espaço vazio para apresentar resultados do lado direito, como pode ser observado na figura 42. Além dessas áreas, esta página possui ainda um menu na barra de navegação superior.

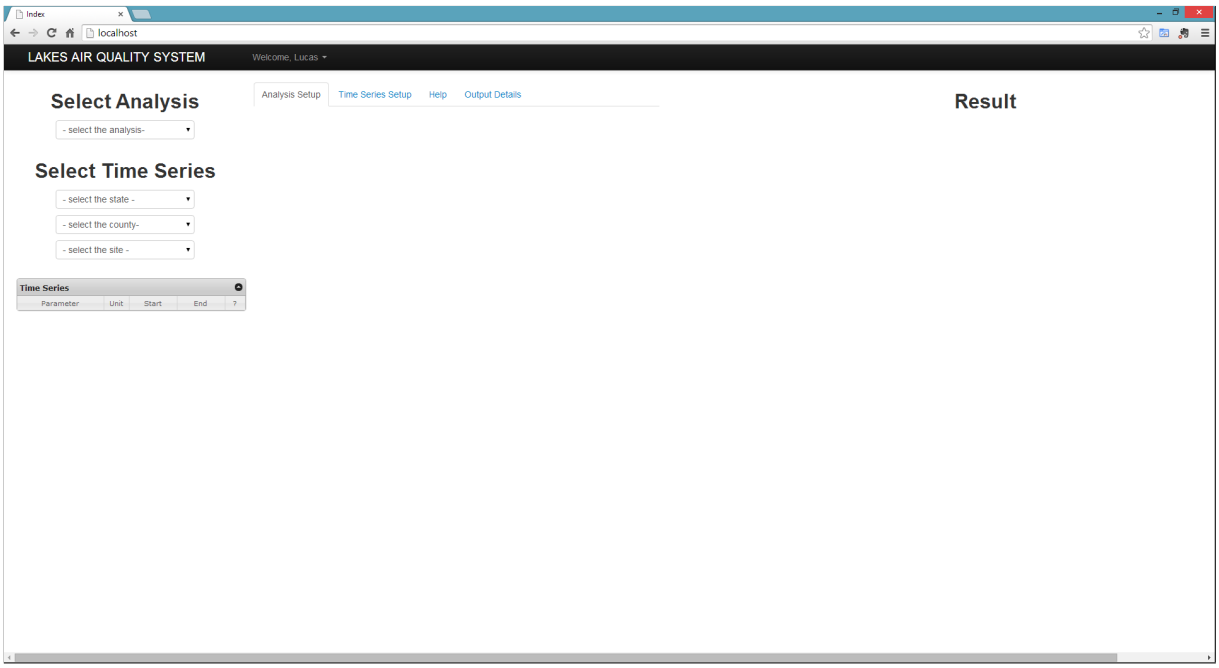

Figura 42: AQS.MVC - Home

Fonte: Autoria Própria

4.4.2 SELECIONANDO ANÁLISES E SÉRIES TEMPORAIS

O menu fixo na parte esquerda da página principal possui diálogos de seleção através dos quais o usuário pode selecionar a análise que ele deseja realizar, bem como as séries temporais que ele deseja utilizar na análise.

Neste momento, nove análises diferentes estão disponíveis para o usuário no painel "*Select Analysis*", o qual contém uma lista com as nove opções, como pode ser observado na Figura 43. Cada opção corresponde a uma das classes implementadas no módulo AQS.R para executar o *script* em R.

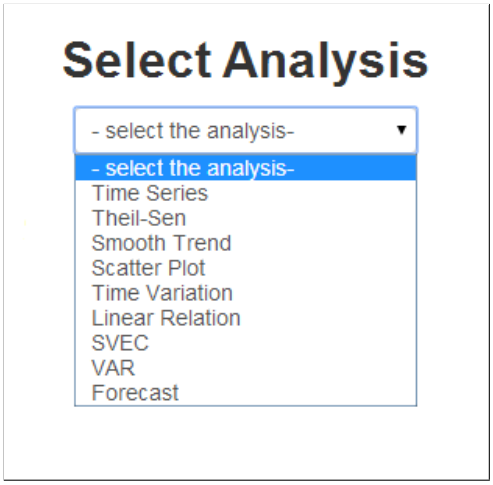

Figura 43: AQS.MVC - Select Analysis

Fonte: Autoria Própria

Para cada uma dessas opções, as abas do painel central serão customizadas. Cada aba apresentará um conteúdo diferente de acordo com as opções disponíveis para cada tipo de análise. Nas próximas seções serão apresentados os conteúdos para cada uma dessas abas.

O segundo painel, ainda no menu fixo do lado esquerdo da página, é utilizado para que o usuário selecione uma ou mais séries temporais, as quais serão utilizadas na análise selecionada no painel apresentado na Figura 43. Este painel possui três listas de seleção, como é possível visualizar na Figura 44.

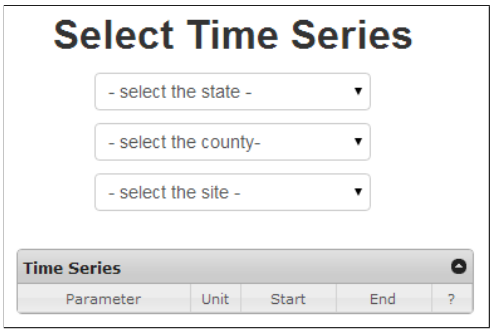

Figura 44: AQS.MVC - Select TimeSeries

Fonte: Autoria Própria

As listas de seleção funcionam em cascata. O usuário precisa selecionar um estado, em seguido um condado e então um sítio para exibir as séries temporais disponíveis para seleção na tabela. Ao selecionar um estado, a lista da condados irá mudar para "- *loading* -", e então para "- *select the county* -"novamente. Esse processo leva alguns segundos, porque acessa a base de

dados para verificar quais condados possuem sítios, evitando que sejam exibidos sítios que não possuem dados na próxima lista de seleção.

Finalmente, depois que o usuário selecionou um estado, um condado e um sítio, a tabela é preenchida com as séries temporais disponíveis naquele sítio, como é possível visualizar na Figura 45. A tabela possui cinco colunas: a coluna "*parameter*", a qual exibe o nome do poluente, a coluna "*unit*", a qual exibe a unidade de medição, as colunas "*start*"e "*end*", as quais exibem quando uma série temporal começou e terminou, respectivamente, e a última coluna, que contém um botão, o qual adiciona aquela série temporal para a lista de séries temporais selecionadas. Uma série temporal não pode ser selecionada mais de uma vez.

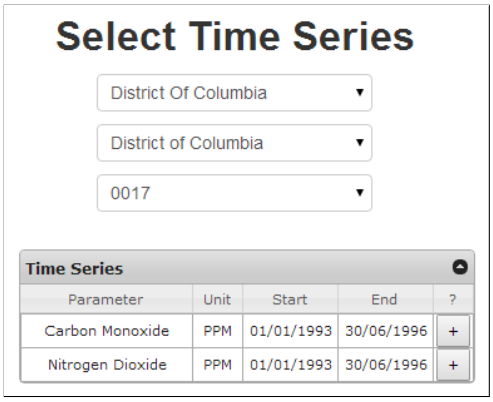

Figura 45: AQS.MVC - Time Series Disponíveis

Fonte: Autoria Própria

Após clicar no botão da última coluna, a série temporal será exibida em outra tabela. Esta nova tabela está disponível na aba "*Selected Time Series*", na parte central da página principal, como pode-se observer na figura 46. Esta tabela tem três colunas a mais do que a tabela utilizada para selecionar a série temporal. Estas colunas exibem o código do estado, o código do condado e o código do sítio, respectivamente.

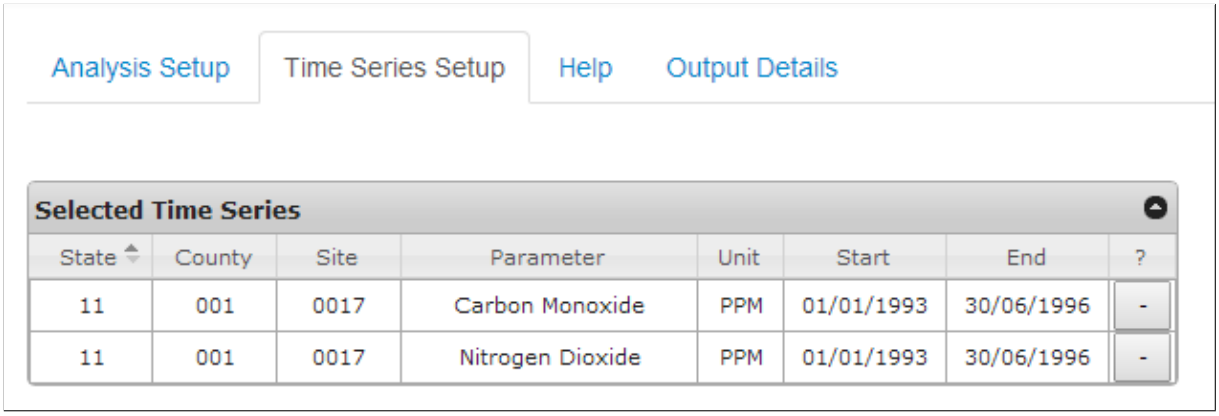

Figura 46: AQS.MVC - Selected Time Setup

Fonte: Autoria Própria

Nesta nova tabela, o usuário pode remover a seleção de uma série temporal a qualquer momento. Para remover a seleção, o usuário só precisa clicar no botão da última coluna da tabela.

4.4.3 CONFIGURANDO E EXECUTANDO UMA ANÁLISE

Depois que o usuário selecionou uma análise e uma série temporal, ele deverá configurar a execução. Como apresentado anteriormente, cada análise possui uma classe em C# que funciona como um *container* para um *script* na linguagem R. Na interface web, cada parâmetro do *script* é apresentado para o usuário em um painel na aba "*Analysis Setup*", no centro da página principal. Adicionalmente aos parâmetros do *script*, o usuário poderá selecionar o período da série temporal que ele deseja utilizar.

4.4.3.1 *TIME SERIES*

A análise "*time series*" desenha a série temporal. O usuário precisa selecionar ao menos uma série temporal para a execução. Como o *script* permite que múltiplas séries temporais sejam utilizadas simultaneamente, uma constante de dez séries temporais foi definida no momento. Se o usuário não selecionar uma série temporal ou se ele selecionar mais do que dez, uma mensagem de erro será exibida. A Figura 47 mostra o painel com os parâmetros disponíveis.

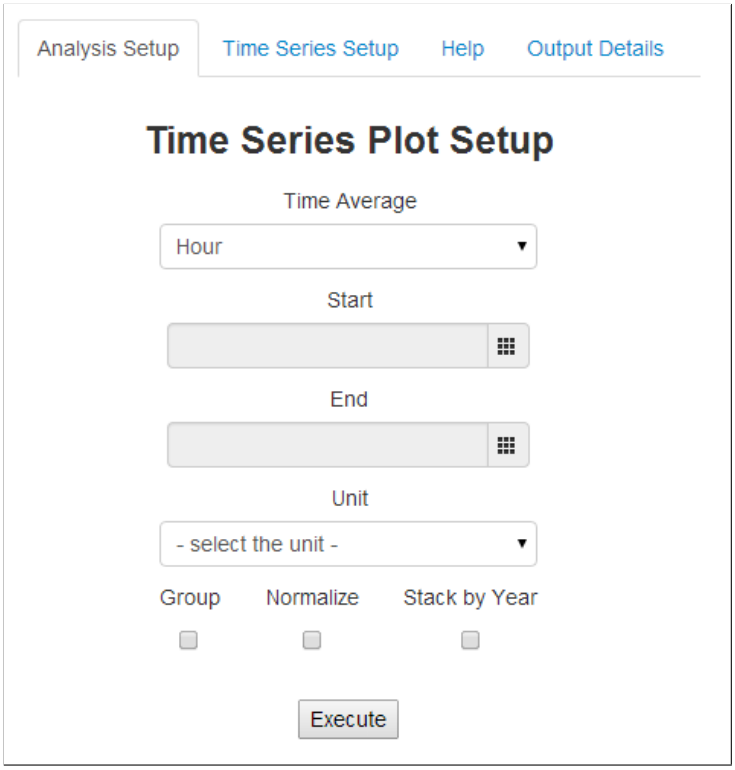

Figura 47: AQS.MVC - Time Series Menu

Fonte: Autoria Própria

Neste painel, o usuário pode selecionar a média na lista de seleção, o qual oferece as opções "*Hour*", "*Day*", "*Month*" e "*Year*". O usuário também pode selecionar as datas de início e fim. Ambos os campos podem ser deixados em branco. Neste caso, a aplicação emprega toda a extensão de datas disponível.

Se o usuário selecionar mais de uma série temporal, ele pode selecionar para agrupá-las, e todas as séries serão desenhadas em um mesmo gráfico. O usuário também pode selecionar se deseja normalizar os dados, o que fará com que as séries temporais sejam, de fato, comparáveis. Finalmente, o usuário pode selecionar se deseja agrupar os dados por ano.

4.4.3.2 *THEIL-SEN*

A função *Theil-Sen* desenha os picos entre todos os pares de pontos, dado um conjunto de n pares (x, y) para uma série temporal. Se o usuário não selecionar uma série temporal, ou selecionar mais do que uma, a aplicação exibe uma mensagem de erro. O painel com os parâmetros é exibido na Figura 48.

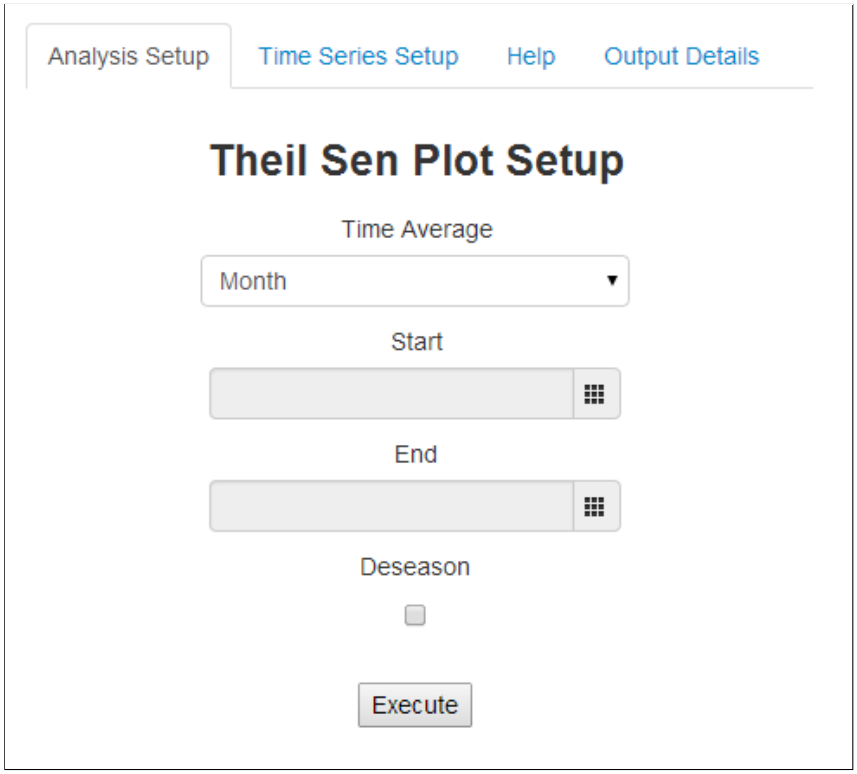

Figura 48: AQS.MVC - Theil-Sen Menu

Fonte: Autoria Própria

Neste painel, o usuário seleciona a média em uma lista, a qual oferece as opções "*Month*", "*Season*", e "*Year*". O usuário também pode selecionar o começo e fim da série temporal. Assim como no caso anterior, ambos os campos podem ser deixados em branco, utilizando toda a extensão de dados disponível. É importante ressaltar que a função Theil-Sen necessita de um período de tempo de seis unidades. Isso significa que se a média selecionada é mensal, a função precisará de no mínimo seis meses. Se a média selecionada for anual, a função precisará de no mínimo seis anos e assim por diante. Finalmente, o usuário poderá selecionar para remover o componente sazonal dos dados.

4.4.3.3 *SMOOTH TREND*

A função *Smooth Trend* desenha as concentrações médias da série temporal selecionada. Pelo menos uma série temporal deverá ser selecionada. Esta função tem os mesmos parâmetros da análise *Theil-Sen*, no entanto, não possui as restrições de período de tempo. O painel é exibido na Figura 49

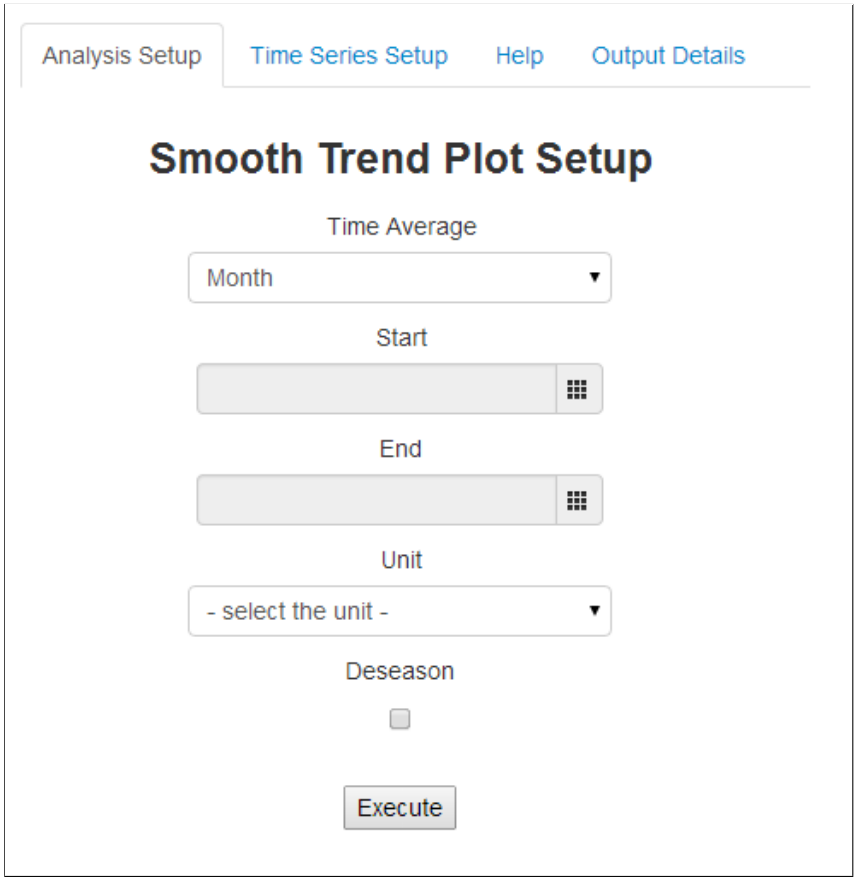

Figura 49: AQS.MVC - Smooth Trend Menu

Fonte: Autoria Própria

4.4.3.4 *SCATTER PLOT*

A análise *Scatter Plot* demonstra o relacionamento entre duas variáveis no mesmo período de tempo. Para executar esta análise, é necessário que o usuário selecione duas séries temporais. Caso contrário, uma mensagem de erro será exibida. Se uma série temporal for maior do que a outra, somente os dados da interseção serão utilizados. Se não houver interseção, a análise não será executada.

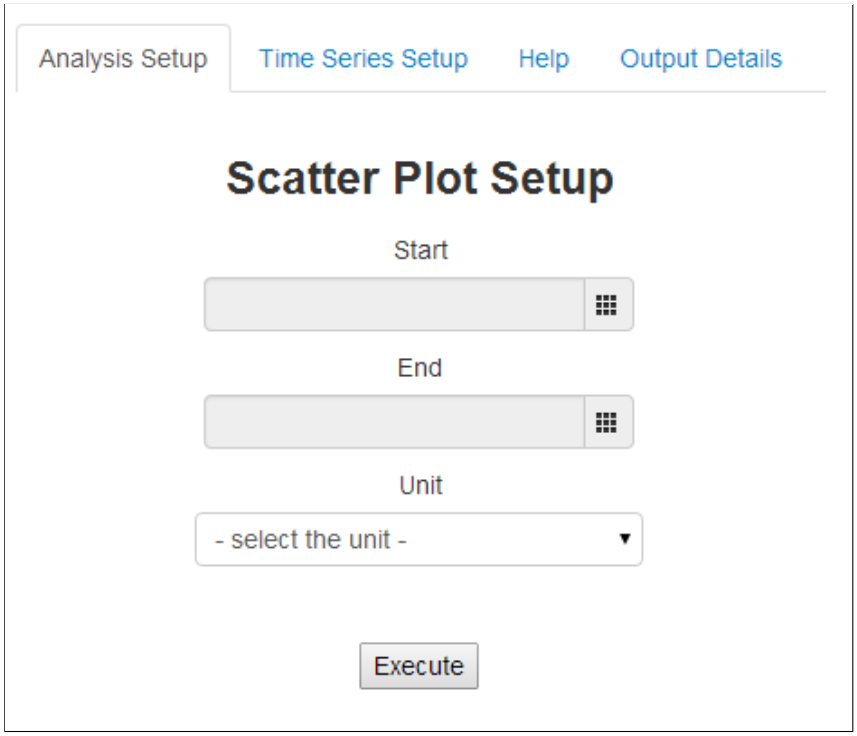

Figura 50: AQS.MVC - Scatter Plot Menu

Fonte: Autoria Própria

4.4.3.5 *TIME VARIATION*

A análise *Time Variation* exibe quatro gráficos diferentes:

1. O primeiro exibe a variação por dia da semana;
2. O segundo exibe a média por hora em um dia;
3. O terceiro exibe a variação durante a semana;
4. O quarto exibe a variação durante os meses.

O usuário poderá selecionar uma ou mais séries temporais. As mesmas opções das funções anteriores estão disponíveis para o usuário, como pode ser observado na Figura 51. No entanto, uma nova opção é "*User Local Time*", para utilizar o horário local na execução do *script*.

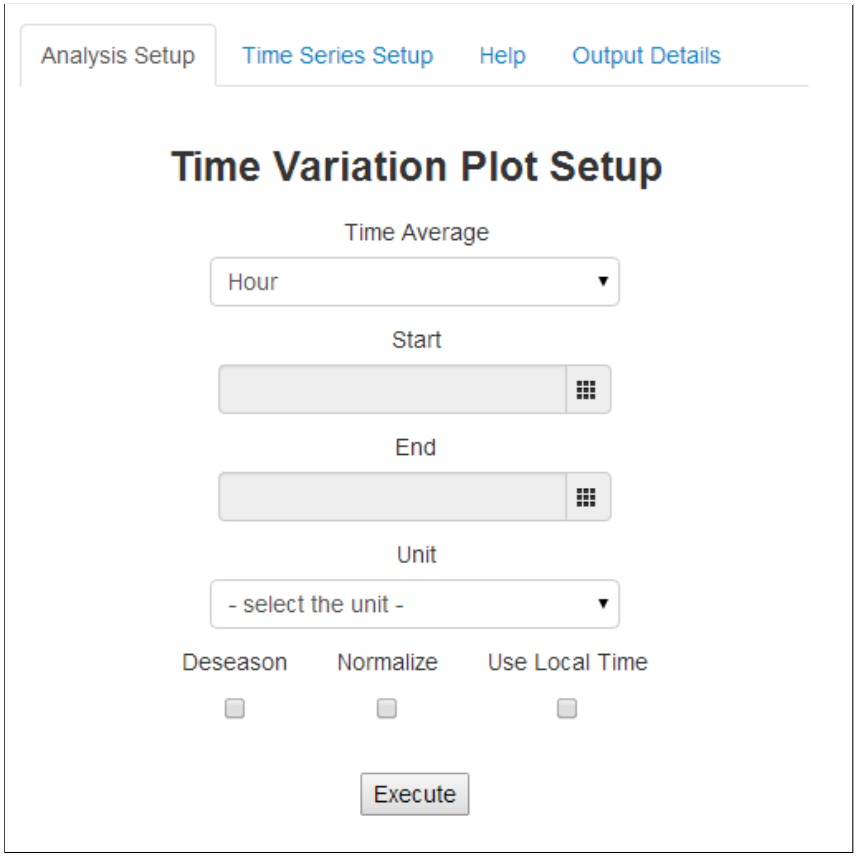

Figura 51: AQS.MVC - Time Variation Menu

Fonte: Autoria Própria

4.4.3.6 *LINEAR RELATION*

A análise *Linear Relation* demonstra a relação linear entre as concentrações de poluentes em duas séries temporais. Assim como na função *Scatter Plot*, somente a interseção será utilizada. No entanto, nesta análise, o usuário pode escolher qual a média que ele deseja utilizar. A Figura 52 mostra o menu para seleção destas opções.

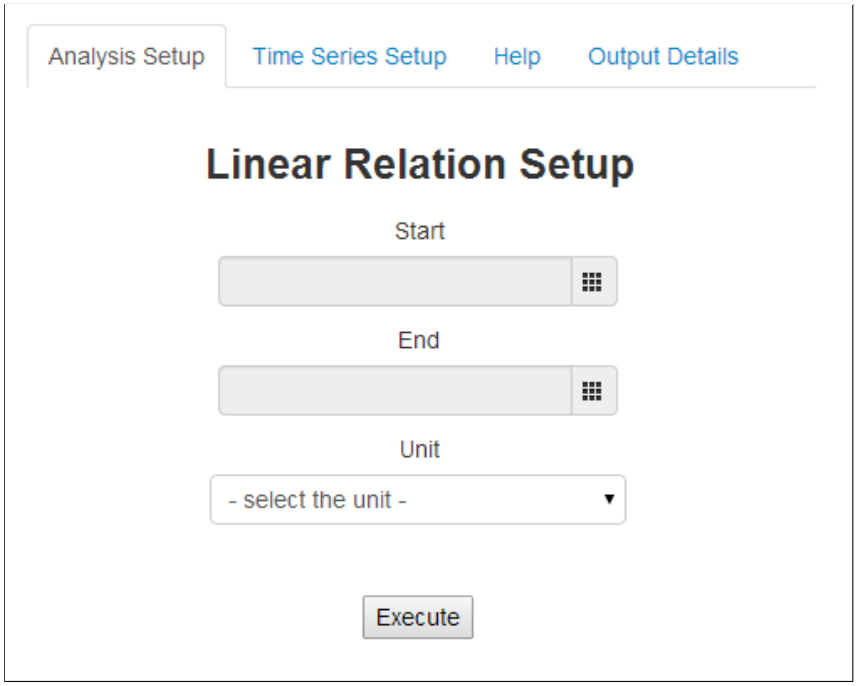

Figura 52: AQS.MVC - Linear Relation Menu

Fonte: Autoria Própria

4.4.3.7 *STRUCTURAL VECTOR ERROR CORRECTION MODEL – SVEC*

A análise do modelo SVEC calcula a cointegração entre duas séries temporais, revelando se as séries temporais apresentam relacionamento a longo prazo. Por tanto, duas séries temporais devem ser selecionadas para a execução desta análise. O usuário pode definir um espaço de tempo se ele desejar utilizar somente um subconjunto dos dados. Esta definição pode ser feita através do menu, ilustrado na Figura 53. O resultado exibirá a resposta de uma variável em relação a outra.

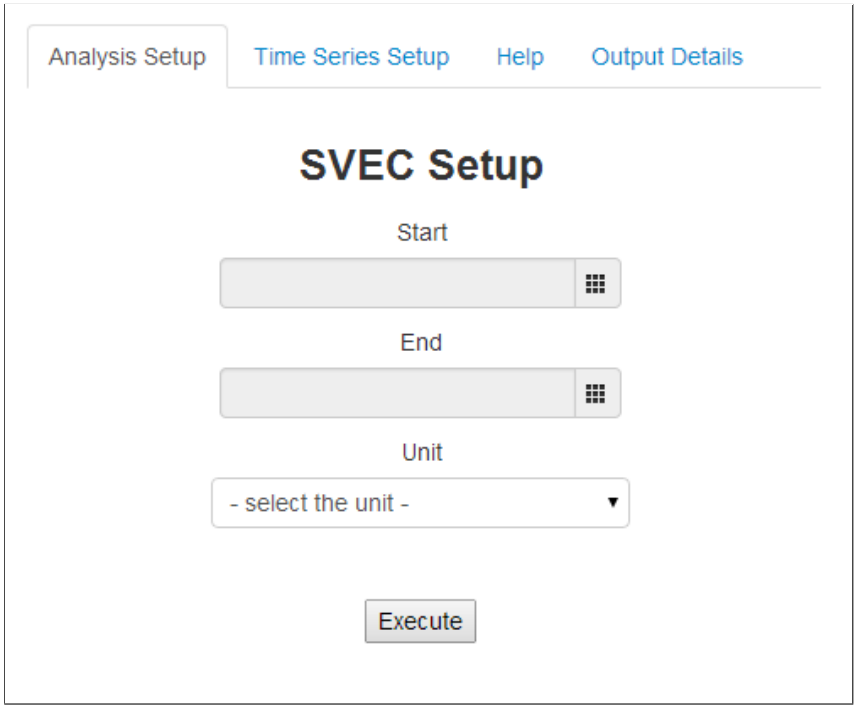

Figura 53: AQS.MVC - SVEC Menu

Fonte: Autoria Própria

4.4.3.8 *VECTOR AUTO REGRESSIVE MODEL – VAR*

Modelos de vetores autoregressivos explicam variáveis endógenas, além dos regressores determinísticos. Baseado em dados do passado, o modelo pode projetar valores futuros. O usuário precisa selecionar pelo menos duas séries temporais para o modelo. Isto se deve ao fato de que o modelo é calculado apenas para a primeira série temporal, utilizando as outras para cálculos do modelo.

Na análise VAR o usuário tem a opção *"Lag"* disponível, como pode ser observado na Figura 54. *Lag* é o número de observações usadas como entrada para o modelo. Este número deve ser maior do que zero. Se o usuário selecionar a opção "*Use Best Lag*", o valor de *Lag* será utilizado como o valor máximo permitido. O melhor *Lag* será calculado de acordo com os métodos Akaike, Hannan-Quinn, Schwarz, and *Forecast Prediction Error*, escolhendo o maior resultado.

O resultado final irá exibir as próximas 24 horas de acordo com o modelo. Se o usuário selecionou um período de tempo e há no banco de dados um período equivalente ao da previsão, uma comparação será exibida.

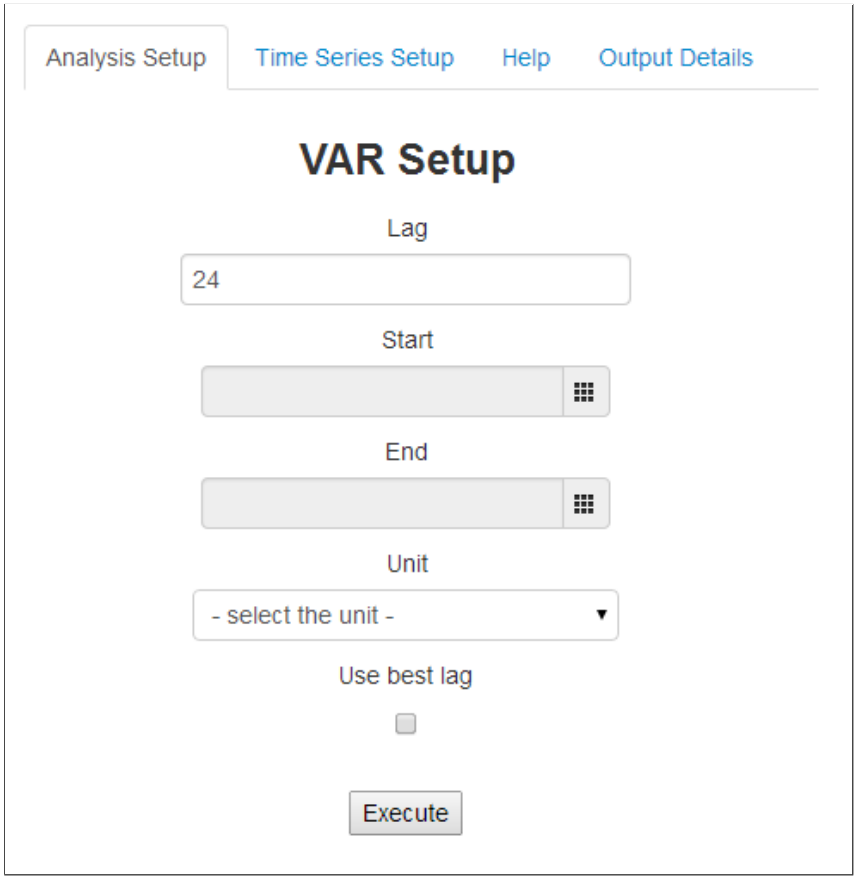

Figura 54: AQS.MVC - VAR Menu

Fonte: Autoria Própria

4.4.3.9 *FORECAST*

A análise *Forecast* prevê dados futuros baseado em um dos quatro métodos disponíveis: ets, arima, rw-drift e naive, os quais foram explicados na seção 4.3. Esta função funciona somente para um série temporal por vez. Além disso, o usuário pode selecionar quantos períodos de tempo que ele deseja prever (ver Figura 55).

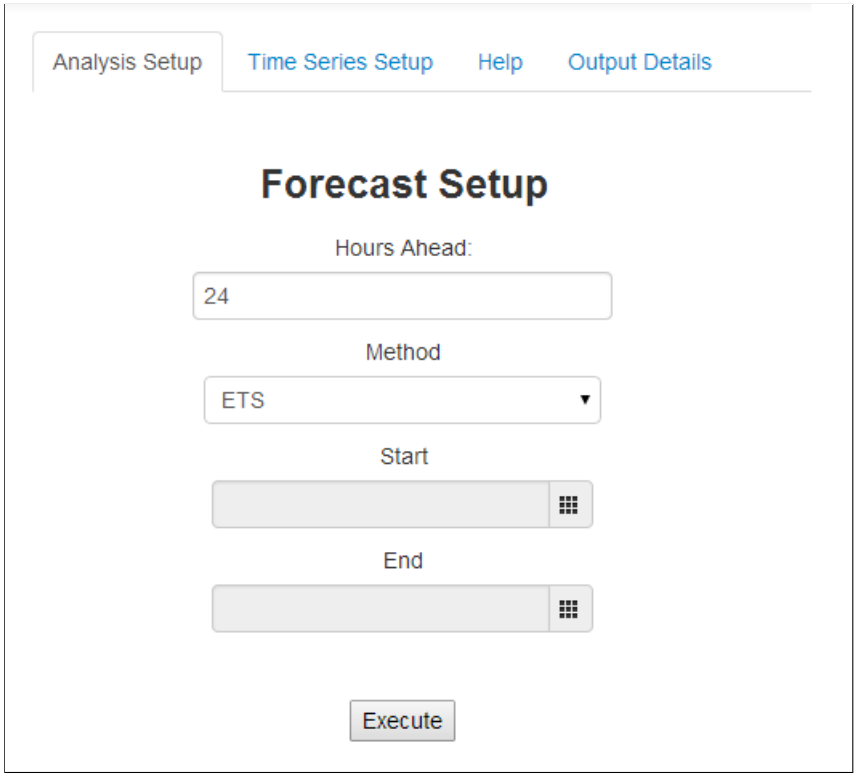

Figura 55: AQS.MVC - Forecast Menu

Fonte: Autoria Própria

4.4.4 AJUDA E DETALHES DE EXECUÇÃO

No painel central da página principal há, além da aba para configuração da análise e da aba que exibe as séries temporais selecionadas que já foram apresentadas nas seções anteriores, as abas "*Help*" e "*Output Details*".

A aba *Help*, ou Ajuda, é utilizada para fornecer ao usuário uma breve explicação sobre a análise que ele selecionou. Por exemplo, se o usuário selecionar a análise "*Time Series*", o texto da aba *Help* irá exibir o número de séries temporais que o usuário deve selecionar, o significado de cada opção e considerações gerais sobre a execução da análise. A Figura 56, mostra esse exemplo.

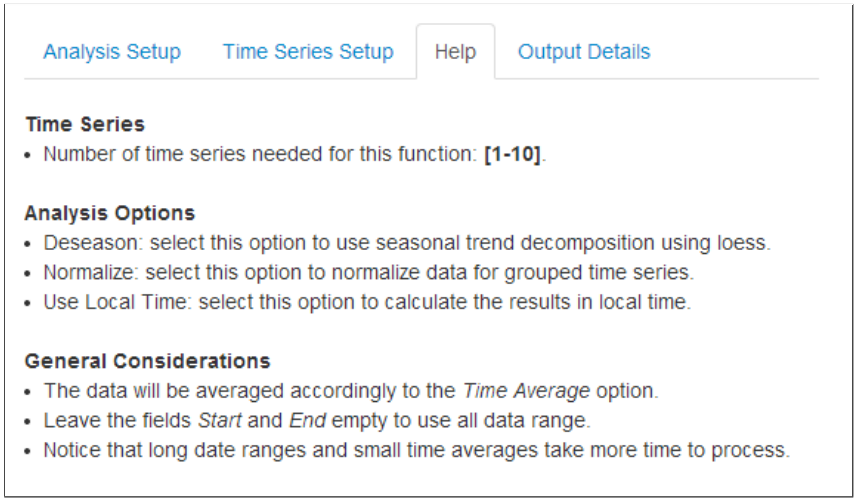

Figura 56: AQS.MVC - Help

Fonte: Autoria Própria

Por fim, a aba *Output Detail*, exibirá links para o *download* do arquivo texto que contém os valores de observação da série temporal e para o arquivo com o gráfico do resultado, como observado na Figura 57.

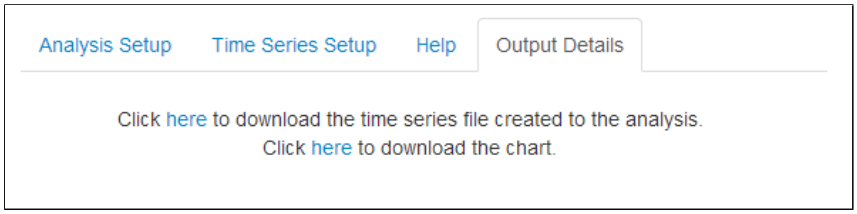

Figura 57: AQS.MVC - Output Details

Fonte: Autoria Própria

4.4.5 RELATÓRIO DE ATIVIDADES E CONFIGURAÇÕES DO USUÁRIO

A última parte da interface do usuário traz, em um menu na barra superior, as opções para exibir as últimas atividades realizadas pelo usuário e editar os dados do usuário, como pode ser visto na Figura 58. Além destas duas opções, o usuário pode escolher encerrar o acesso ao sistema.

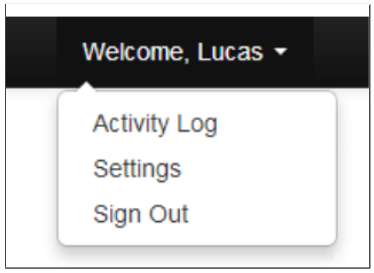

Figura 58: AQS.MVC - Barra de Navegação
Fonte: Autoria Própria

Se o usuário selecionar a opção "*Activity Log*", ele será redirecionado para um página onde poderá consultar até as cinco últimas análises que realizou. Para cada análise será exibido, como pode ser observado na Figura 59, o nome da análise, a data e hora em que foi realizada, os dados das séries temporais utilizadas, e links para o arquivo texto (com os dados utilizados) e para a imagem (com o gráfico resultante).

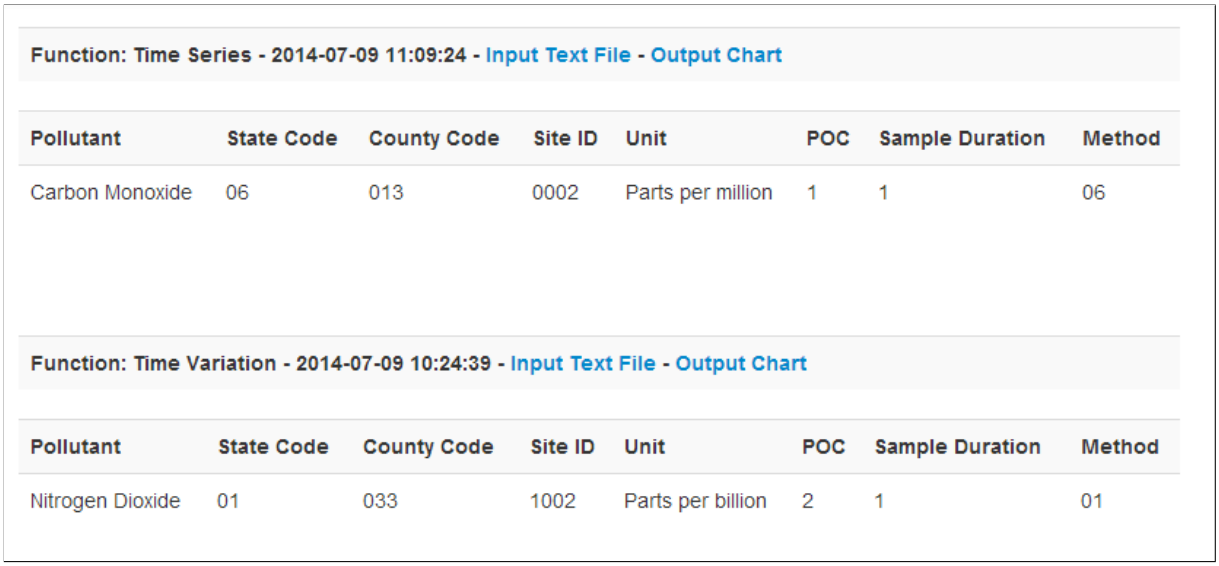

Figura 59: AQS.MVC - Activity Log
Fonte: Autoria Própria

Por fim, se o usuário selecionar a opção "*Settings*", será redirecionado para uma página onde poderá alterar os dados do seu cadastro, como exibido na Figura 60, seguindo os mesmos critérios de quando o cadastro foi criado pela primeira vez.

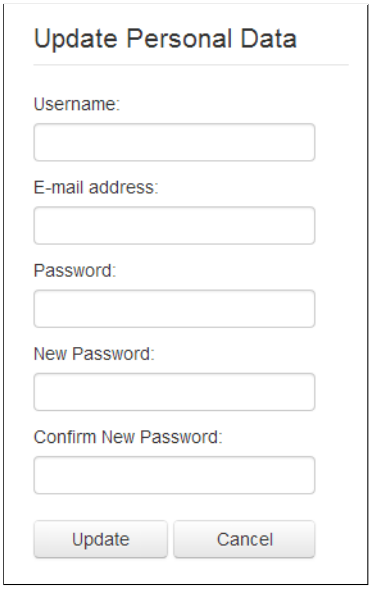

Figura 60: AQS.MVC - Settings

Fonte: Autoria Própria

4.4.6 FECHAMENTO

A utilização de uma interface web para que os usuários acessem as análises permite que a computação seja distribuída. O servidor web, a base dados e o processador da linguagem R podem estar em máquinas diferentes. Isso permite que a base de dados cresça e que mais memória seja adicionada para o armazenamento, se necessário. No entanto, essa distribuição é transparente para o usuário final.

5 ESTUDOS DE CASO

Para demonstrar a utilização do sistema para que novos usuários possam, no futuro, ter um guia de referência, bem como desenvolver uma prova de conceito, neste capítulo são exibidos estudos de caso, utilizando o sistema para buscar conclusões estatísticas e ambientais.

5.1 ESTUDO DE CASO 1 - UTILIZANDO UMA SÉRIE TEMPORAL

Utilizando apenas uma série temporal, o usuário terá acesso à análises para séries temporais univariadas. No entanto, este fator não diminui a validade dos resultados obtidos. O primeiro passo é escolher a série temporal que será utilizada. Para abranger um volume maior de dados, a série temporal escolhida será uma série de monóxido de carbono, as quais possuem, em geral, dados coletados a cada hora, diferentemente das séries de chumbo ou PM2.5, as quais possuem, em geral, dados coletados a cada mês.

A série temporal específica para este estudo de caso foi coletada no estado do Arizona, no condado de Maricopa, no *site* 13. Ela contém dados coletados do dia 13 de fevereiro de 1996 até 30 de abril de 1999, medidos em partes por milhão.

A escolha de uma série temporal é possível pela manipulação de dados através da utilização do MongoDB. Ao escolher um estado, o banco de dados filtra os resultados para exibir apenas os condados que contêm *sites* com séries temporais, evitando que o usuário fique procurando *sites* que contêm dados. Esta operação demora alguns poucos segundos e percorre toda a base de dados através de índices.

Para verificar o comportamento da série temporal e realizar análises mais significativas, inicia-se a análise com a execução do *script Time Series*. Para este *script*, seleciona-se a média horária e a unidade parte por milhão. O resultado, exibido na Figura 61 demonstra que existe sazonalidade aditiva: no meses de janeiro acontecem os valores mais altos da série e nos meses de julho, os mais baixos.

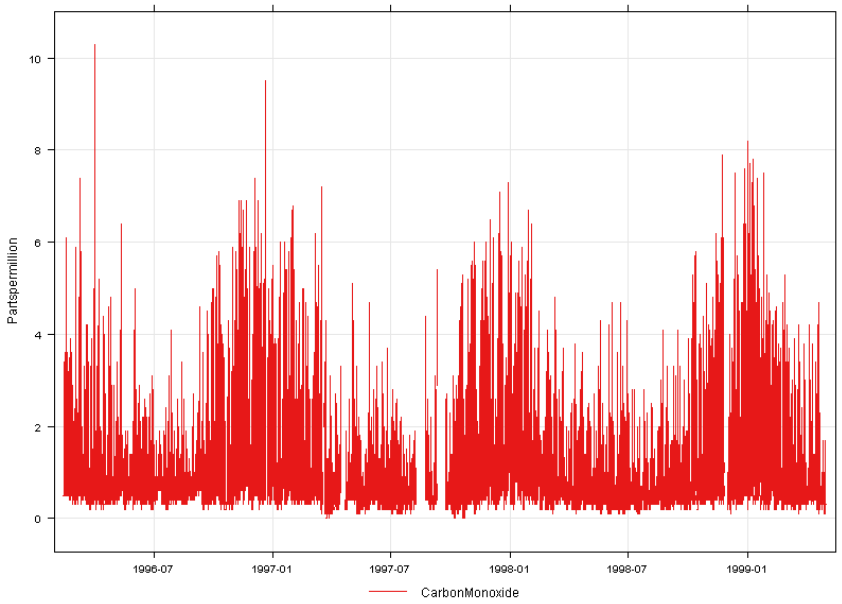

Figura 61: Estudo de Caso 1 - Série Temporal

Fonte: Autoria Própria

Para confirmar a existência de sazonalidade, executa-se a análise *Smooth Trend*. O resultado é exibido na Figura 62.

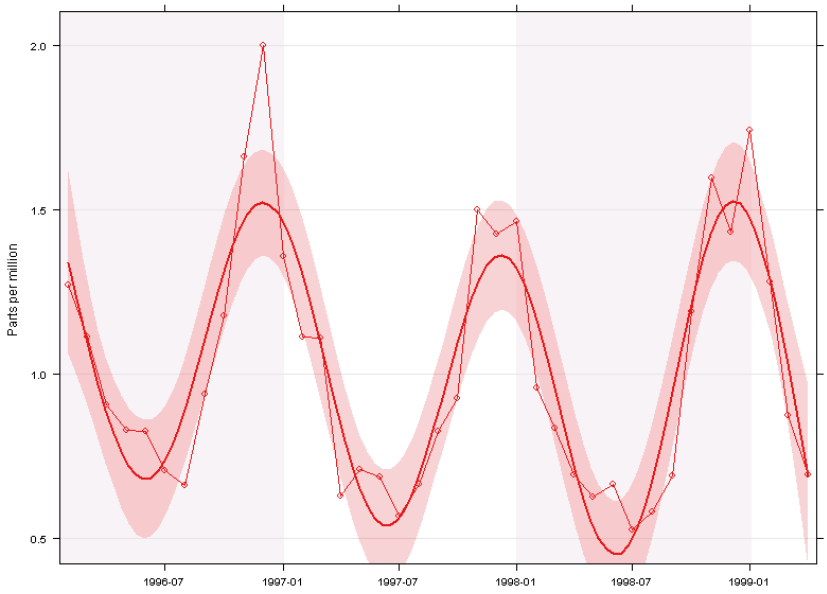

Figura 62: Estudo de Caso 1 - *Smooth Trend*

Fonte: Autoria Própria

Observando-se os valores máximos da análise *Smooth Trend*, não é possível obter conclusões. No entanto, observando-se os valores mínimos, tem-se a impressão de que existe uma tendência de decrescimento. Executando a análise *Theil-Sen* é possível verificar que esta tendência existe. Ela pode ser observada na Figura 63.

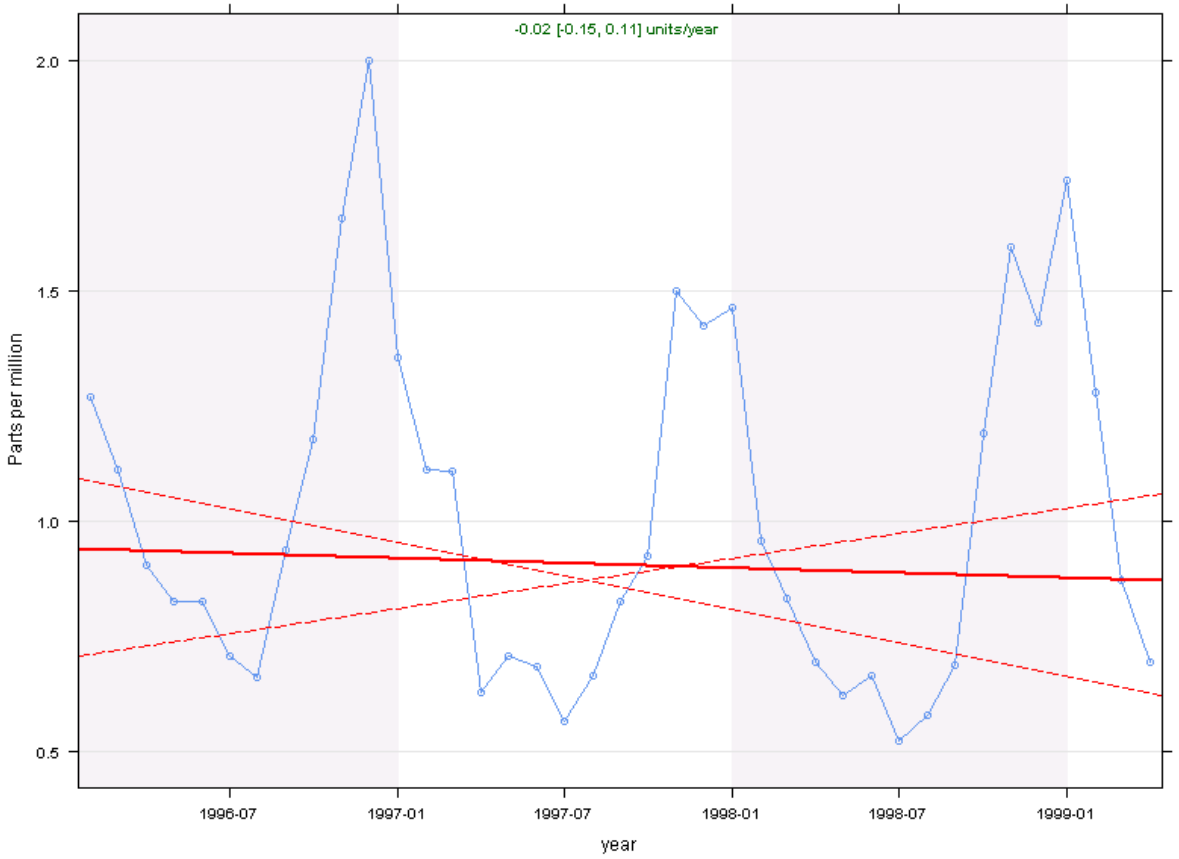

Figura 63: Estudo de Caso 1 - *Theil-Sen*

Fonte: Autoria Própria

Finalmente, realiza-se a análise *Time Variation*. Esta análise mostra que existem, em média, dois picos durante um dia: um pela manhã e um pela noite, como pode ser observado na Figura 64. Ainda segundo o resultado, confirma-se a sazonalidade e, por fim, verifica-se que existem picos nos finais de semana. A quebra da análise segundo estes critérios, afim de remover máximos ou isolar períodos de um dia para obter resultados mais significativos, ainda não é possível através do sistema, porém, se desejar, o usuário pode obter os dados para realizar estas análises independentemente.

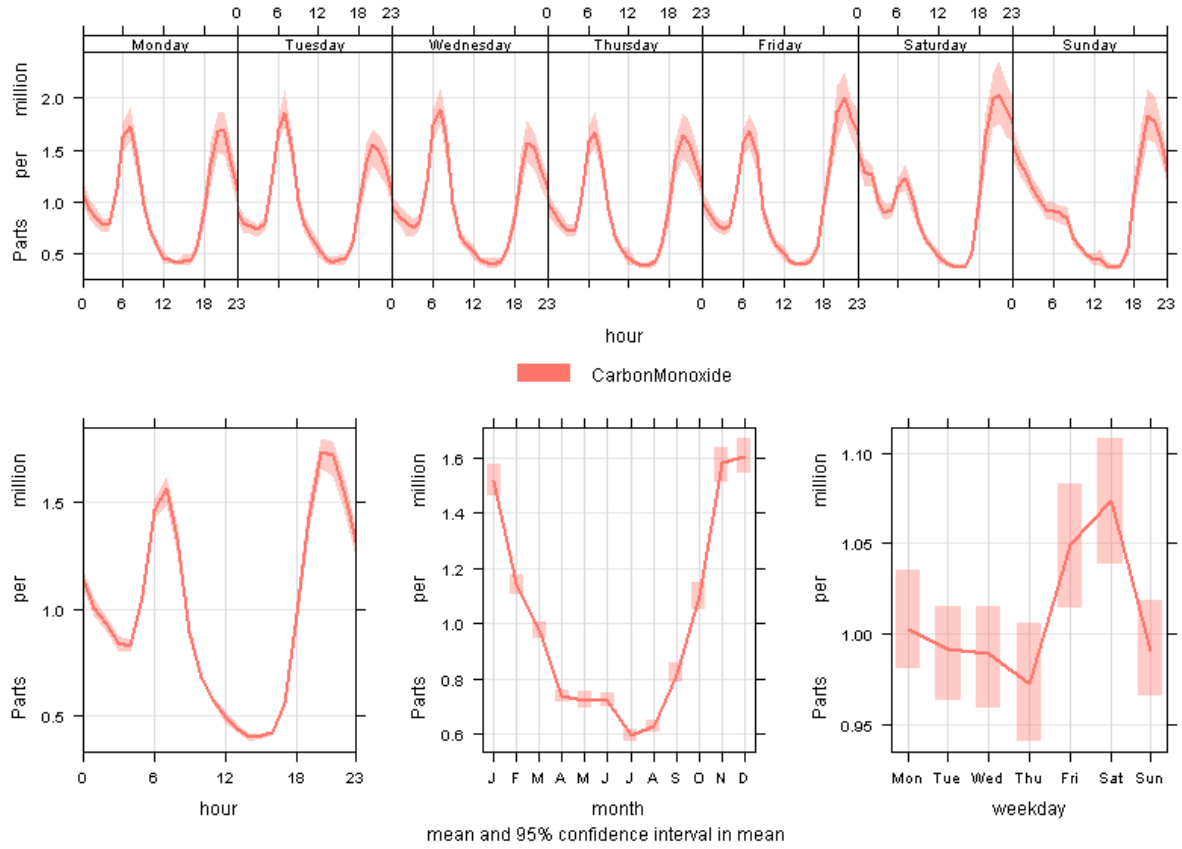

Figura 64: Estudo de Caso 1 - *Time Variation*

Fonte: Autoria Própria

5.2 ESTUDO DE ASO 2 - UTILIZANDO DUAS SÉRIES TEMPORAIS

Os resultados obtidos utilizando apenas uma série temporal podem ser significativos, porém exigem séries temporais maiores e mais recentes para que seja possível chegar a conclusões ambientais e não somente estatísticas.

Com isso em mente, é possível selecionar duas séries temporal, uma de dióxido de nitrogênio e outra de ozônio, por exemplo, os quais se relacionam, pelo fato do ozônio ser produzido, em grande parte, pela interação entre dióxido e monóxido de nitrogênio com compostos orgânicos voláteis.

No estado *District of Columbia*, no *site* 0017, existem séries temporais para esses gases no mesmo período de tempo, o qual inicia-se em primeiro de janeiro de 1993 e termina em 30 de junho de 1996.

Executando a análise *Time Series* para visualizar as séries temporais, é possível verificar, como exibido na Figura 65, que não há dados para a séries de ozônio no ano de 1994. Portanto, será utilizado o período referente aos anos de 1995 e 1996.

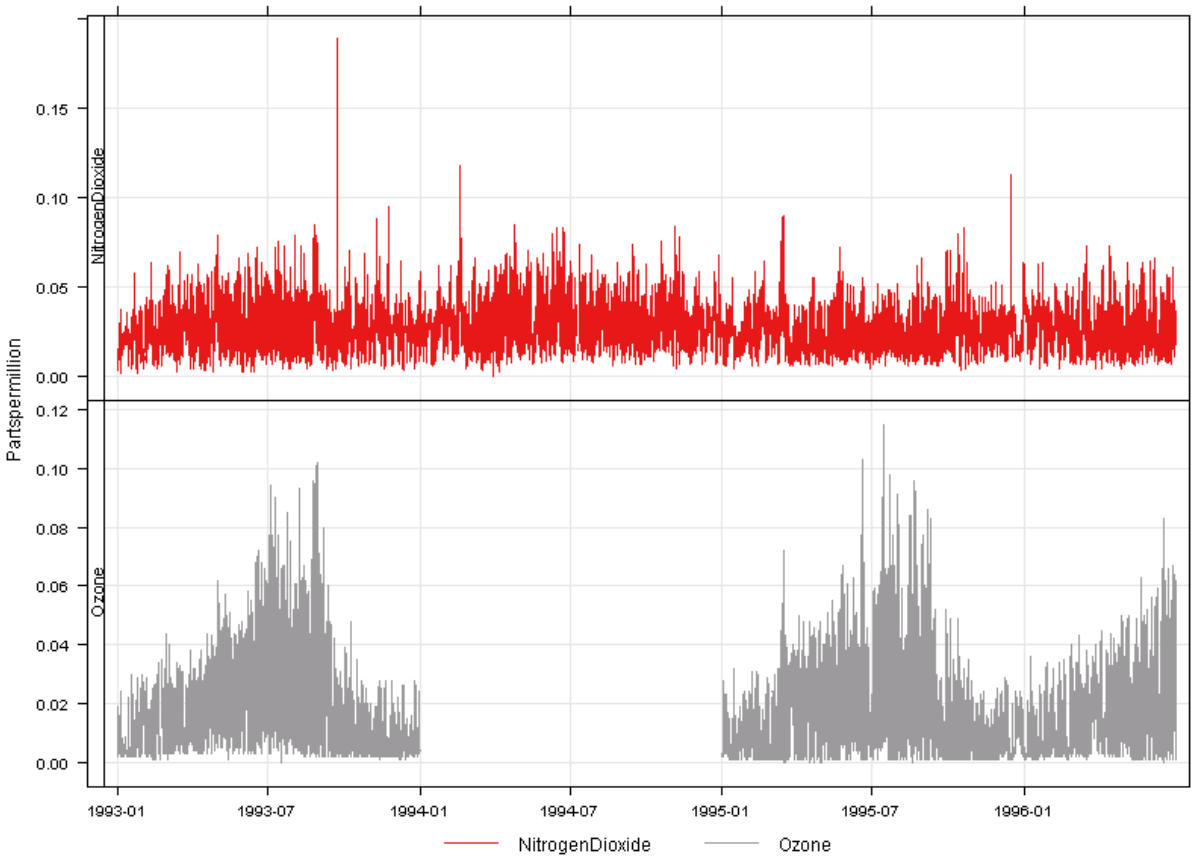

Figura 65: Estudo de Caso 2 - Série Temporal

Fonte: Autoria Própria

Afim de verificar a relação entre as duas variáveis, executa-se a análise *Scatter Plot*. Esta análise mostra que NO2 e O3 possuem uma correlação negativa, ou seja, quando um aumenta, o outro diminui, como pode ser observado na Figura 66.

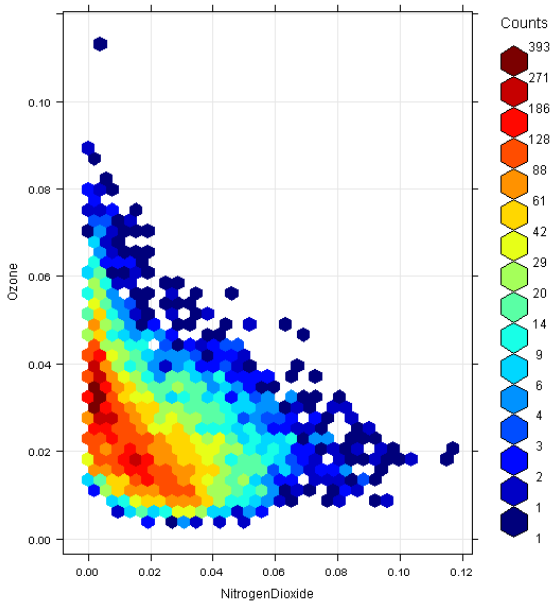

Figura 66: Estudo de Caso 2 - *Scatter Plot*

Fonte: Autoria Própria

Também é possível visualizar a relação linear entre as duas séries temporais através da análise *Linear Relation*, como pode ser observado na Figura 67.

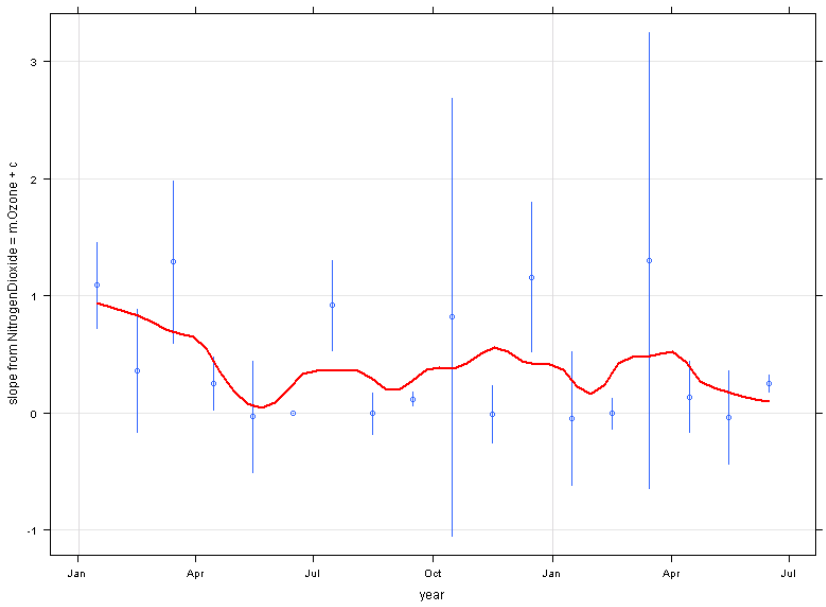

Figura 67: Estudo de Caso 2 - *Linear Relation*

Fonte: Autoria Própria

5.3 ESTUDO DE CASO 3 - PREVISÃO

Por fim, é possível realizar a previsão de dados futuros utilizando os dados existentes na base de dados. O sistema oferece dois tipos de previsão de dados: *forecast* e VAR, como apresentados anteriormente. O primeiro é utilizado para a previsão de séries temporais univariadas e o segundo para séries multivariadas.

Uma série temporal que apresenta um comportamento interessante é a série de Ozônio do *site* 0119, em Navajo, Arizona. Como apresentado na Figura 68, que mostra o resultado da análise Theil-Sen, esta série possuiu uma característica de crescimento. Aplicando-se, então, a análise *forecast*, com o parâmetro *ets*, espera-se que a previsão seja realizada através de um amortecimento aditivo. O resultado é exibido na Figura 69.

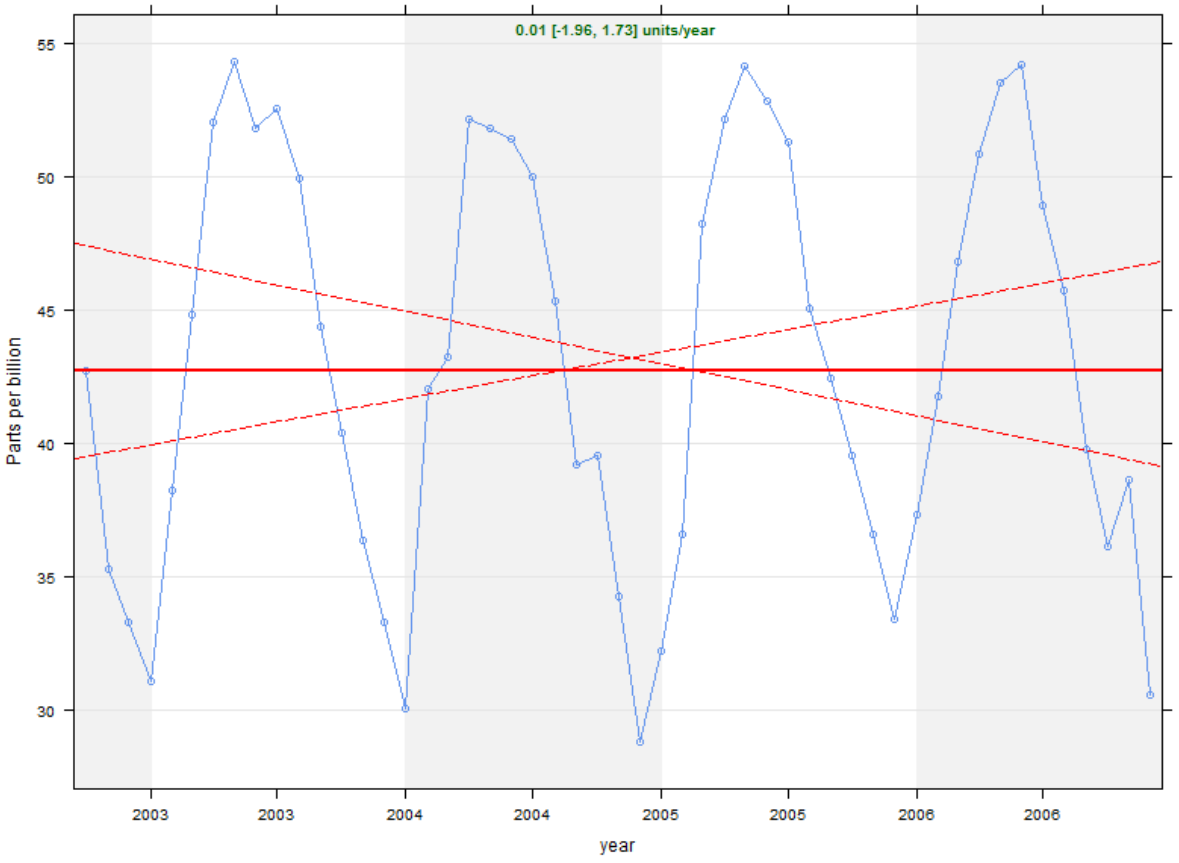

Figura 68: Estudo de Caso 3 - *Theil-Sen*

Fonte: Autoria Própria

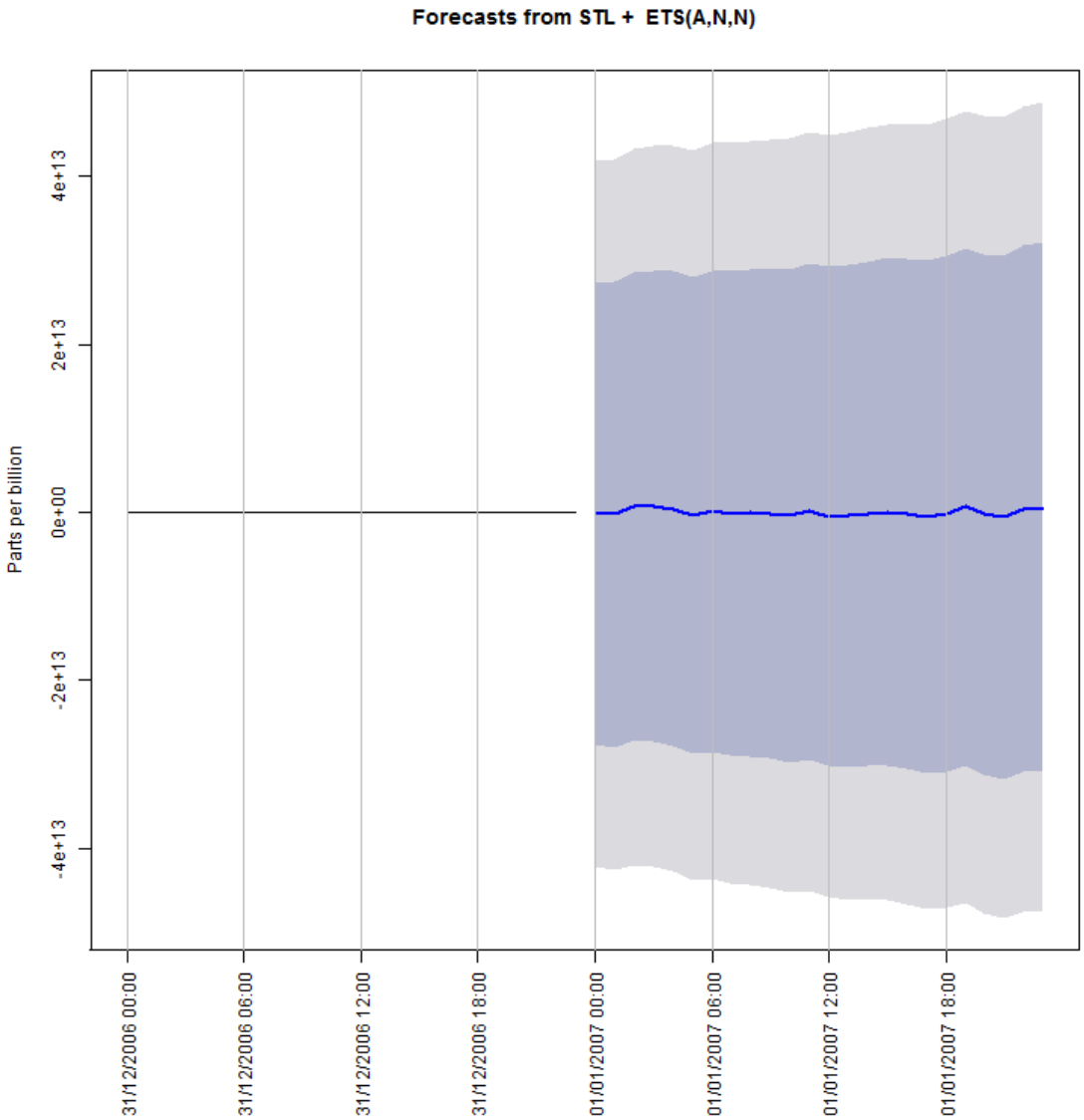

Figura 69: Estudo de Caso 3 - *Forecast*

Fonte: Autoria Própria

Se o usuário deseja saber se há relacionamento entre os níveis de ozônio e algum outro poluente, pode, então, executar a análise SVEC. Está análise demonstrará o comportamento de um poluente em relação ao outro. Por exemplo, no site 002, em Aleutians East, no Alaska, temos uma série de partículas de bromo e outra de partículas de cálcio que, no ano de 2004, relacionaram-se como demonstrado no Figura 70. A previsão pela análise VAR, portanto, pode ser realizada, obtendo o resultado da Figura 71, a qual mostra valores de bromo para as próximas 24 horas, levando em consideração ambas as séries.

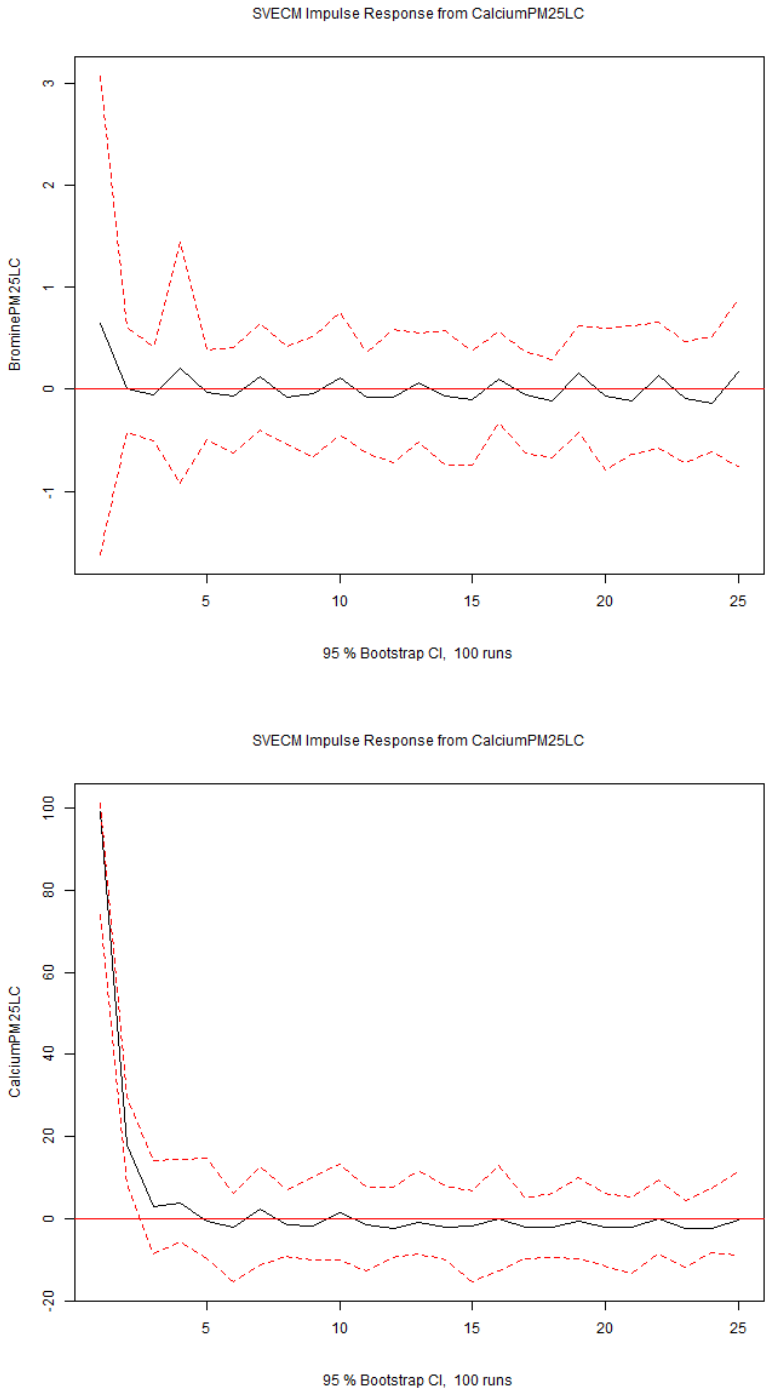

Figura 70: Estudo de Caso 3 - *SVEC*

Fonte: Autoria Própria

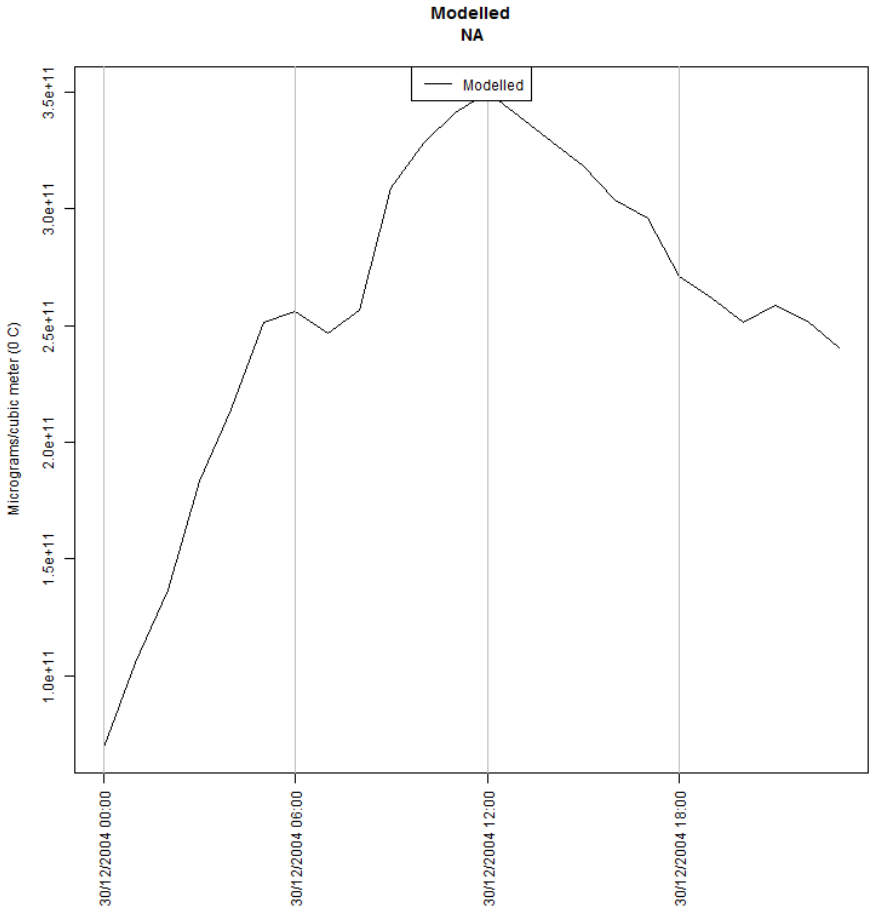

Figura 71: Estudo de Caso 3 - *VAR*

Fonte: Autoria Própria

5.3.1 TRANSFORMAÇÃO DOS DADOS

Análises mais profundas podem ser realizadas por especialistas das áreas ambientais e/ou da área estatística. No entanto, os estudos de caso mostram a capacidade do sistema em organizar, acessar e processar um grande volume de dados, dando significado a eles através de gráficos e da organização dos dados em séries temporais.

6 CONCLUSÃO

Análise de dados é um campo do conhecimento há muito tempo estudado pela ciência da computação. Nos últimos anos, com o desenvolvimento dos recursos de *hardware*, o volume de dados criado se tornou muito maior do que a capacidade de processamento dos *softwares* existentes em algumas áreas. Esta lacuna abriu espaço para o desenvolvimento da *Big Data*.

Apesar de ser uma área muito grande e que engloba diferentes métodos e metodologias, a *Big Data* tem sido muito importante para a análise de dados. Para dados referentes as ciências atmosféricas não é diferente. Com acesso a uma grande base de dados aberta, como a base de dados referente à qualidade do ar do AQS, é preciso saber manipular os dados para que conclusões sejam obtidas quanto ao significado dos dados.

A fase inicial deste trabalho possibilitou a compreensão dos sistemas para *Big Data*. Em seguida, a análise e o entendimento dos dados disponíveis viabilizou a sua utilização em um sistema para analizá-los.

A última versão da especificação do sistema desenvolvido apresentava nove requisitos, os quais necessitavam que quatro módulos distintos fossem desenvolvidos para o seu funcionamento. Todos os módulos foram implementados e todos os requisitos atendidos na versão atual do sistema. Em alguns casos a interpretação dos requisitos foi expandida, assegurando que ela seria atendida de todas as formas possíveis. Por exemplo, a exigência para que o usuário fosse capaz de visualizar o resultado da análise, foi expandida para que o usuário também seja capaz de visualizar o resultado de análises passadas.

A partir da criação do sistema, estatísticos e especialistas das áreas ambientais, principalmente aqueles que trabalham com gerenciamento de poluentes, poderão utilizá-lo para compreender melhor os dados disponíveis pela EPA. Além disso, a forma como o trabalho foi realizado, sendo construído em módulos, permitirá que estas mesmas pessoas desenvolvam seus próprios trabalhos e as insiram no sistema para testá-los. Estes trabalhos podem ser sobre tecnologias NoSQL, funções MapReduce ou ainda cálculos estatísticos utilizando a linguagem de programação R.

O sistema, neste momento, encontra-se apoiado diretamente sobre banco de dados MongoDB, possuindo propriedades NoSQL para armazenar e utilizar os dados disponíveis. Sendo este banco de dados uma ferramenta *open source*, ele pode ser estudado e modificado para que se torne mais eficiente para os tipos de dados específicos utilizados e para séries temporais. Este banco de dados também oferece meios para implementar funções MapReduce, o que pode melhorar o desempenho de algoritmos específicos. Por último, para adicionar um *script* em R para a solução, basta que o usuário crie uma classe que funcione como um *container* para os parâmetros necessários para a sua execução.

Por fim, a interface web é bastante simples e foi desenvolvida como um protótipo afim de facilitar a compreensão do sistema e de suas funcionalidades. No futuro ela deverá sofrer alterações para tornar a experiência do usuário ainda melhor.

Este sistema, portanto, como um produto de dados, pode agregar valor aos dados, transformando-os e exibindo-os de diferentes formas. A seguir são apresentados os caminhos que podem ser seguidos para dar continuação a este trabalho e ao estudo da *Big Data*.

6.1 TRABALHOS FUTUROS

Este trabalho abre oportunidades para novos trabalhos em termos de pesquisa, utilizando o sistema como apoio, e em termos de desenvolvimento, aplicando refinamentos ao sistema para que ele se torne um produto completo.

Como suporte à pesquisa, tem-se:

- Estudo da eficiência na recuperação de dados utilizando MongoDB.

  O MongoDB é um sistema *open-source* e, portanto, pode ser customizado para que se torne um sistema apropriado para análise de séries temporais. Os conceitos aplicados ao seu desenvolvimento podem ser estudados e melhorados para que a indexação, a distribuição, a recuperação e o processamento de dados de séries temporais, em específico, sejam mais eficientes.

- *Benchmark* de tecnologias SQL vs NoSQL.

  Como as tecnologias NoSQL são recentes, ainda não há muitos estudos referentes a sua real eficiência. A modularização do sistema permite que um módulo para um sistema de gerenciamento de banco de dados baseado em SQL seja criado e que os resultados, utilizando ambos os módulos, sejam comparáveis.

- Estudo e criação de algoritmos MapReduce para melhorar a eficiência da aplicação.

  O MongoDB possuiu um sistema próprio para execução de tarefas MapReduce. Além disso, também é possível integrá-lo com outras plataformas como o Hadoop. Com a importação dos dados, tornou-se possível pensar em tarefas MapReduce para extração de dados estatísticos da base de dados.

- Criação de scripts em R para análises mais sofisticadas.

  A modularização do sistema permite que um usuário crie seu próprio *script* em R, bem como as classes necessárias para incorporá-lo como um componente ao sistema existente, ampliando ainda mais as possibilidades de análises que o sistema pode executar.

Como melhoria ao sistema, tem-se:

- Alteração da camada de acesso aos dados, para utilização de outras bases de dados;
- Refatoração dos módulos, aumentando a independência de cada parte;
- Melhora da interface da solução, adicionando mapas e outras propriedades gráficas;
- Capacidade de aceitar arquivos de séries temporais do usuário;
- Oferecer ao usuário os modelos resultantes das previsões para que possam ser utilizados sobre novos dados.

Estes são apenas alguns dos trabalhos que ainda podem ser executados sobre este empreendimento para que ele traga mais resultados para os interessados e cumpra todos os papéis imaginados no início do seu desenvolvimento. Este trabalho é, necessariamente, um projeto em contínuo progresso.

REFERÊNCIAS

Air Quality System. 2014. Disponível em: <http://www.epa.gov/ttn/airs/airsaqs/>. Accesso em: 25 ago. 2014.

AQS Codes and Descriptions. 2014. Disponível em: <http://www.epa.gov/ttn/airs/airsaqs/\\manuals/codedescs.htm>. Accesso em: 18 mai. 2014.

AQS Data Dictionary. 2011. U.S. Environmental Protection Agency, Office of Air Quality Planning and Standards, Information Transfer and Program Integration Division, Information Management Group Disponível em: <http://www.epa.gov/ttn/airs/airsaqs/manuals/AQS\%20Data\%20Dictionary.pdf>. Accesso em: 18 mai. 2014.

ASP.NET MVC4. 2014. Disponível em: <http://www.asp.net/mvc/mvc4>. Accesso em: 08 jul. 2014.

BEGOLI, E. 2012. A Short Survey on the State of the Art in Architectures and Platforms for Large Scale Data Analysis and Knowledge Discovery from Data. **Proceedings of the WICSA/ECSA 2012 Companion Volume.** Helsinki, Finland, p. 177-183. Disponível em: <http://doi.acm.org/10.1145/2361999.2362039>. Accesso em: 25 ago. 2014.

Binary JSON. 2014. Disponível em: <http://bsonspec.org>. Accesso em: 07 jul. 2014.

BOEHM, B. W. 1988. A Spiral Model of Software Development and Enhancement. **Computer.** IEEE Computer Society Press. Los Alamitos, CA, USA v. 21, n. 5. p.61-72. ISSN 0018-9162. Disponível em: <http://dx.doi.org/10.1109/2.59>. Accesso em: 25 ago. 2014.

BUSHMAN, F. D. et al. 2013. Bringing it all together: big data and HIV research. **Wolters Kluwer Health.** Lippincott Williams and Wilkins, v. 27, n. 5, p. 835-838. ISSN 0269-9370.

C# and .NET MongoDB Driver. 2014. Disponível em: <http://docs.mongodb.org/ecosystem/drivers/csharp/>. Accesso em: 07 jul. 2014.

CARSLAW, D. C.; ROPKINS, K. 2012. The openair manual — open-source tools for analysing air pollution data. Manual for version 1.0, King's College London. Disponível em: <http://www.openair-project.org/PDF/OpenAir_Manual.pdf>. Accesso em: 25 ago. 2014.

CASTILLO, C. 2005. Effective Web Crawling. **ACM Special Interest Group on Information Retrieval** New York, NY, USA v. 39, n. 1. p.55-56. ISSN 0163-5840. Disponível em: <http://doi.acm.org/10.1145/1067268.1067287>. Accesso em: 25 ago. 2014.

CHANG, F. et al. 2008. Bigtable: A Distributed Storage System for Structured Data. **ACM Transactions on Computer Systems**. New York, NY, USA, v. 26, n. 2, p. 4:1-4:26. ISSN 0734-2071. Disponível em: <http://doi.acm.org/10.1145/1365815.1365816>. Accesso em: 25 ago. 2014.

CHAUDHURI, S.; WEIKUM, G. 2000. Rethinking Database System Architecture: Towards a Self-Tuning RISC-Style Database System. **Proceedings of the 26th International Conference on Very Large Data Bases.** Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, USA, p. 1-10. Disponível em: <http://dl.acm.org/citation.cfm?id=645926.671696>. Accesso em: 25 ago. 2014.

CHEN, H.; CHIANG, R. H. L.; STOREY, V. C. 2012. Business Intelligence and Analytics: From Big Data to Big Impact. **MIS Quarterly.** Society for Information Management and The Management Information Systems Research Center. Minneapolis, MN, USA, v. 36, n. 4, p. 1165-1188. ISSN 0276-7783. Disponível em: <http://dl.acm.org/citation.cfm?id=2481674.2481683>. Accesso em: 25 ago. 2014.

CHO, J.; GARCIA-MOLINA, H.; WIDOM, J. 2001. Crawling the Web: Discovery and Maintenance of Large-Scale Web Data. Disponível em: <http://oak.cs.ucla.edu/~cho/papers/cho-thesis.pdf>. Accesso em: 25 ago. 2014.

DEAN, J.; GHEMAWAT, S. 2012. MapReduce: Simplified Data Processing on Large Clusters. **Communications of the ACM.** New York, NY, USA, v. 51, n. 1, p. 107-113. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1327452.1327492>. Accesso em: 25 ago. 2014.

Download Detail AQS Data. 2014. Disponível em: <http://www.epa.gov/ttn/airs/airsaqs\\/detaildata/downloadaqsdata.htm>. Accesso em: 18 mai. 2014.

FISHER, D. et al. 2012. Interactions with Big Data Analytics. **ACM Interactions.** New York, NY, USA, v. 19, n. 3, p. 50-59. ISSN 1072-5520. Disponível em: <http://doi.acm.org/10.1145/2168931.2168943>. Accesso em: 25 ago. 2014.

FORECAST, FORECASTING FUNCTIONS FOR TIME SERIES AND LINEAR MODELS. 2014. Disponível em: <http://cran.r-project.org/web/packages/forecast/index.html>. Accesso em: 07 jul. 2014.

Gain vital insight from your data. 2013. Package 'forecast'. Disponível em: <http://public.dhe.ibm.com/common/ssi/ecm/en/imw14717usen/IMW14717USEN.PDF>. Accesso em: 11 ago. 2014.

GALLAGHER, S. 2012. How IBM's Deep Thunder delivers "hyper-local" forecasts 3-1/2 days out. **ArsTechnica.** Disponível em: <http://goo.gl/tl3rPC>. Accesso em: 25 ago. 2014.

GHEMAWAT, S.; GOBIOFF, H.; LEUNG, S.-T. 2003. The Google File System. **Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles.** Bolton Landing, NY, USA, p. 29-43. Disponível em: <http://doi.acm.org/10.1145/945445.945450>. Accesso em: 25 ago. 2014.

GUO, Z.; FOX, G.; ZHOU, M. 2012. Investigation of Data Locality and Fairness in MapReduce. **Proceedings of Third International Workshop on MapReduce and Its Applications Date.** Delft, The Netherlands, p. 25-32. Disponível em: <http://doi.acm.org/10.1145/2287016.2287022>. Accesso em: 25 ago. 2014.

Hadoop. 2013. Disponível em: <https://hadoop.apache.org/>. Accesso em: 25 ago. 2014.

HAMM, S. 2013. How Big Data Can Boost Weather Forecasting. **Wired.** Disponível em: <http://www.wired.com/insights/2013/02/how-big-data-can-boost-weather-forecasting/>. Accesso em: 25 ago. 2014.

HYNDMAN, R. J. 2014. Package 'forecast'. Disponível em: <http://cran.r-project.org/web/packages/forecast/forecast.pdf>. Accesso em: 04 ago. 2014.

JOHNSTON, S. 2013. WeatherSignal: Big Data Meets Forecasting. **Scientific American.** Disponível em: <http://blogs.scientificamerican.com/guest-blog/2013/10/11/weathersignal-big-data-meets-forecasting/>. Accesso em: 25 ago. 2014.

KANG, Y.; KUNG, S. H.; JANG, H.-J. 2013. Simulation Process Support for Climate Data Analysis. **Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference.** Miami, Florida, USA, p. 29:1-29:6. Disponível em: <http://doi.acm.org/10.1145/2494621.2494651>. Accesso em: 25 ago. 2014.

KNAPP, A. 2013. Forecasting the Weather With Big Data And The Fourth Dimension. **Forbes.** Disponível em: <http://www.forbes.com/sites/alexknapp/2013/06/13/forecasting-the-weather-with-big-data-and-the-fourth-dimension/>. Accesso em: 25 ago. 2014.

KOLBERG, W. et al. 2013. MRSG - A MapReduce Simulator over SimGrid. **Parallel Computing.** Elsevier Science Publishers B. V. Amsterdam, The Netherlands, The Netherlands v. 39, n. 4-5. p.233-244. ISSN 0167-8191. Disponível em: <http://dx.doi.org/10.1016/j.parco.2013.02.001>. Accesso em: 25 ago. 2014.

Lakes Environmental. 2014. Disponível em: <http://www.weblakes.com/index.html/>. Accesso em: 25 ago. 2014.

LAKSHMAN, A.; MALIK, P. 2010. Cassandra: A Decentralized Structured Storage System. **ACM Special Interest Group on Operating Systems**. New York, NY, USA, v. 44, n. 2, p. 35-40. ISSN 0163-5980. Disponível em: <http://doi.acm.org/10.1145/1773912.1773922>. Accesso em: 25 ago. 2014.

LOUKIDES, M. 2010. What is data science?. **O'Reilly radar.** Disponível em: <http://radar.oreilly.com/2010/06/what-is-data-science.html>. Acesso em: 25 ago. 2014.

MARX, V. 2013. The Challenges of Big Data. **Nature.** Macmillian Publishers Limited, v. 498, p. 255-260. Disponível em: <http://www.nature.com/nature/journal/v498/n7453/full/498255a.html>. Accesso em: 25 ago. 2014.

MICHNA, P.; WOODS, M. 2013. RNetCDF – A Package for Reading and Writing NetCDF Datasets. **The R Journal.** v. 5, n. 2. p.29-37. Disponível em: <http://journal.r-project.org/archive/2013-2/michna-woods.pdf>. Accesso em: 25 ago. 2014.

MongoDB. 2014. Disponível em: <http://www.mongodb.org>. Accesso em: 27 jun. 2014.

MURDOCH, T. B.; DETSKY, A. S. 2013. The Inevitable Application of Big Data to Health Care. **The Journal of American Medical Association**, v. 309, n. 13, p. 1351-1352. Disponível em: <http://jama.jamanetwork.com/article.aspx?articleid=1674245>. Accesso em: 25 ago. 2014.

MvcJqGrid. 2014. Disponível em: <https://www.nuget.org/packages/MvcJqGrid/>. Accesso em: 08 jul. 2014.

OpenAir. 2014. Disponível em: <http://www.openair-project.org/>. Accesso em: 25 ago. 2014.

OpenSignal. 2013. Disponível em: <http://opensignal.com/>. Accesso em: 25 ago. 2014.

PATIL, D. 2012. Data Jujitsu: The art of turning data into product. **O'Reilly radar.** Disponível em: <http://radar.oreilly.com/2012/07/data-jujitsu.html>. Acesso em: 25 ago. 2014.

SAX, C.; STEINER, P. 2013. Temporal Disaggregation of Time Series. **The R Journal.** v. 5, n. 2. p.80-88. Disponível em: <http://journal.r-project.org/archive/2013-2/sax-steiner.pdf>. Accesso em: 25 ago. 2014.

SELTZER, M. 2008. Beyond Relational Databases. **Communications of the ACM.** New York, NY, USA, v. 51, n.7, p. 52-58. Disponível em: <http://doi.acm.org/10.1145/1364782.1364797>. Accesso em: 25 ago. 2014.

SHAFER, I. et al. 2013. Specialized Storage for Big Numeric Time Series. **Proceedings of the 5th USENIX Conference on Hot Topics in Storage and File Systems.** USENIX Association. San Jose, CA, USA, p. 15-15. Disponível em: <http://dl.acm.org/citation.cfm?id=2534861.2534876>. Accesso em: 25 ago. 2014.

SHANG, H. 2013. Ftsa: An R Package for Analyzing Functional Time Series. **The R Journal.** v. 5, n. 1. p.64-73. Disponível em: <http://journal.r-project.org/archive/2013-1/shang.pdf>. Accesso em: 25 ago. 2014.

STONEBRAKER, M. et al. 2007. One Size Fits All? - Part 2: Benchmarking Results. **Proceedings of the Third International Conference on Innovative Data Systems Research (CIDR).** Disponível em: <http://nms.csail.mit.edu/~stavros/pubs/osfa.pdf>. Accesso em: 25 ago. 2014.

STONEBRAKER, M.; CETINTEMEL, U. 2005. "One Size Fits All": An Idea Whose Time Has Come and Gone. **Proceedings of the 21st International Conference on Data Engineering.** IEEE Computer Society. Washington, DC, USA, p. 2-11. Disponível em: <http://dx.doi.org/10.1109/ICDE.2005.1>. Accesso em: 25 ago. 2014.

STONEBRAKER, M. et al. 2007. The End of an Architectural Era: (It's Time for a Complete Rewrite). **Proceedings of the 33rd International Conference on Very Large Data Bases.** VLDB Endowment. Vienna, Austria, p.1150-1160 Disponível em: <http://dl.acm.org/citation.cfm?id=1325851.1325981>. Accesso em: 25 ago. 2014.

Support for selecting array elements in return specifier. 2014. Disponível em: <https://jira.mongodb.org/browse/SERVER-828>. Accesso em: 07 jul. 2014.

The R Project for Statistical Computing. 2014. Disponível em: <http://www.r-project.org/>. Accesso em: 25 ago. 2014.

The United States Environmental Protection Agency. 2014. Disponível em: <http://www.epa.gov/>. Accesso em: 25 ago. 2014.

Twitter Bootstrap. 2014. Disponível em: <http://getbootstrap.com/2.3.2/>. Accesso em: 08 jul. 2014.

VARS: VAR MODELLING. 2014. Disponível em: <http://cran.r-project.org/web/packages/vars/index.html>. Accesso em: 07 jul. 2014.

Yahoo! Search Webmap. 2013. Disponível em: <http://goo.gl/SGK6xM>. Accesso em: 25 ago. 2014.

ZHENG, Y.; LIU, F.; HSIEH, H.-P. 2013. U-Air: When Urban Air Quality Inference Meets Big Data. **Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.** Chicago, Illinois, USA, p. 1436-1444. Disponível em: <http://research.microsoft.com/pubs/193973/U-Air-KDD-camera-ready.pdf>. Accesso em: 25 ago. 2014.