Novas Ferramentas para Visualização Georreferenciada de Dados: uma integração entre R e Google Maps

Luís Gustavo Silva e Silva

Dissertação apresentada
ao
Departamento de Estatística
da
Universidade de Federal de Minas Gerais
para
obtenção do título
de
Mestre em Ciências

Programa: Programa de Pós-graduação em Estatística
Orientador: Prof. Dr. Renato Martins Assunção
Coorientador: Prof. Dr. Marcelo Azevedo Costa

Belo Horizonte, fevereiro de 2012

Novas Ferramentas para Visualização Georreferenciada de Dados: uma integração entre R e Google Maps

Esta versão da dissertação contém as correções e alterações sugeridas pela Comissão Julgadora durante a defesa da versão original do trabalho, realizada em 15/02/2013.

Comissão Julgadora:

- Prof. Dr. Renato Martins Assunção (orientador) - UFMG
- Prof. Dr. Marcelo Azevedo Costa (coorientador)- UFMG
- Prof. Dr. Francisco José de Azevêdo Cysneiros - UFPE
- Prof. Dr. Marcos Oliveira Prates - UFMG
- Prof. Dr. Danilo Lourenço Lopes - UFMG

Agradecimentos

Agradeço primeiramente a Deus, pela minha saúde e paz de espírito concedido ao longo de toda minha vida.

Aos meus pais, Dorcina e Wilson, pelo amor e carinho transmitido em todos esses anos longe deles. Papai, com toda sua experiência de vida, que pode ser notada pelo olhar, agradeço por sempre ter me passado a mensagem de buscarmos a felicidade e não esquecermos de nossas origens. Mamãe, pelo seu sorriso contagiante e pela sua alegria de viver. Aos meus irmãos, Pedro e Carol, meus melhores amigos.

Aos meus familiares, em especial aos meus tios Zezinho e Carlinhos, pelo incentivo incansável ao longo dos meus estudos.

A minha namorada Priscila, por estar ao meu lado durante todo o mestrado, acreditando nos meus sonhos e vivendo todos eles comigo. Obrigado pela sua compreensão, carinho, amor, amizade e todas as alegrias que você já me proporcionou.

Aos amigos de Governador Valadares, Bruno pela sua alegria, presteza e, claro, por ser a pessoa que tem o pensamento mais rápido para fazer uma piada. Meu cunhado Guilherme, por ser uma figura inspiradora. Lucas Braga, pelo amizade sincera e por me fazer pensar em uma distribuição de probabilidades para as palavras que ele diz. Marcelo, por proporcionar gargalhadas de fazer chorar. Marconi, por sempre acreditar e incentivar meus estudos na estatística, desde de 2004. Roberto, pela amizade sincera e pelos tempos de graduação.

Agradeço ao meus amigos da graduação, em especial ao Samuel, amigo de muita pureza e de uma caridade invejável. Bruno, por sempre cair nas piadas e por ter sido um grande parceiro nos estudos. Iago, amigo que levarei para sempre ao meu lado, muito obrigado por ser meu testador de software da madrugada. Laura, é sinônimo de alegria, badalação, e claro, de uma amizade inigualável.

Aos amigos do mestrado, em especial, aos amigos gaúchos, Fernanda, Laura e Rodrigo, e mineira Paola. Fernanda, pela sua amizade e por ser uma grande parceira no desenvolvimento do meu trabalho de mestrado. Rodrigo, pela sua entrega nos nossos problemas e pela parceria no Expedito. Laura, pelos inúmeros almoços de domingo, e claro que continuarei filando essa comidinha. Paola, por ter aprendido ser uma pessoa mais dedicada, vendo-a estudando todos os dias, e claro pela sua alegria de viver.

Ao meus orientadores, Marcelo e Renato, por confiarem no meu trabalho e pelo aprendizado ao longo do mestrado.

Agradeço as funcionárias da secretária de pós-graduação, Rogéria e Rose, pelas conversas, risadas e pelo incentivo. Agradeço a Maiza, pelo cafézinho nosso de cada dia, sem ele este trabalho não teria sido completado. Ao porteiro Black, que foi o primeiro funcionário da UFMG que conheci, homem de muita alegria, simpatia e simplicidade.

Aos colegas do Grupo São Tomé, em especial ao Edré, por sempre tirar minhas dúvidas computacionais e por testar todas as funções do pacote.

Bring what is needed to ***Solve*** *the Problem.*

George Casella

Resumo

Neste trabalho apresentamos o desenvolvimento e as funcionalidade do pacote `aRouca` . O pacote implementa novas visualizações georreferenciadas aplicadas à avaliação de dados de profissionais de saúde. A principal contribuição deste trabalho é a integração de linguagens e ambientes de visualização, como as linguagens `SVG` , `KML` , API's do Google e a linguagem R . Foram desenvolvidos novos recursos como a visualização de gráficos de barras, gráficos de setores e imagens integradas à linguagem `KML` . Em particular, foram utilizados recursos do Google Charts API para a geração dinâmica de imagens como chamadas de url em arquivos `KML` . O pacote `aRouca` é utilizado junto à plataforma Arouca, um banco de dados que agrega informações sobre os profissionais de saúde que atuam no Sistema Único de Saúde (SUS), e permite ao usuário a visualização dinâmica e interativa do fluxo de profissionais de saúde ao longo do território nacional.

Palavras-chave: Software R , Google Earth/Maps, visualização de dados, KML, SVG.

Abstract

This paper presents the development and functionalities of the aRouca package. The package proposes new visualizations of geocoded data and was originally created to explore data of health professionals. The main contribution of this work is the integration of visualization environments and languages such as SVG, KML, Google APIs and R. We developed new features as viewing bar charts, pie charts and images integrated into the KML language. In particular, we use Google Charts API for generating dynamic images as url in KML files. The package works integrated into the Arouca platform, a database system that aggregates information about health professionals who work in the Brazilian National Health System, and it allows the user to view dynamic and interactive flow of health professionals throughout nationwide.

Keywords: Software R, Google Earth/Maps, Data Visualization, KML, SVG.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Considerações Preliminares

Dados são fundamentais no âmbito da ciência, pois é através destes que obtemos conhecimento sobre a natureza de um fenômeno. A coleta de dados, em particular, depende de cada experimento, como por exemplo, na educação, onde são utilizados testes para tentar mensurar a habilidade dos alunos em alguma disciplina. Neste exemplo, o instrumento de coleta dos dados são os testes, enquanto que os dados são as respostas dos alunos.

A organização dos dados é uma etapa importante para extrairmos informações. Uma ferramenta amplamente utilizada são os computadores, uma vez que estes fornecem ferramentas eficazes no armazenamento e exploração dos dados. Esta forma de armazenamento permite vários tipos de alocações, como tabelas (matrizes) simples e bidimensionais, ambas muito utilizadas e são casos particulares de bancos de dados, cujo conceito é mais abrangente, ver Ullman e Widom (1997).

Métodos para coleta e análise de dados evoluíram ao longo dos últimos 200 anos, para se tornar a base de uma das vertentes da matemática, que conhecemos hoje como estatística. Esses métodos podem fornecer uma visão profunda e compreensão sobre os processos e fenômenos subjacentes dos dados. A aplicação destes métodos pode aumentar o valor e utilidade dos dados, além de sugerir como proceder no futuro para obter dados adicionais que poderiam ser ainda mais úteis (Unwin *et al.*, 2006; Young *et al.*, 2006).

Novos tipos e estruturas de dados foram surgindo ao longo do tempo, e algumas técnicas

de análise estatística tiveram que surgir para se adequar a estas novas estruturas de dados. Além disso, novas técnicas foram desenvolvidas para simplificar e melhorar o processo de análise de dados. Dentre estas técnicas está a visualização de dados, que foi impulsionada também pelos avanços computacionais nas últimas décadas.

Segundo Friedman (2008), o principal objetivo da visualização de dados é comunicar a informação de forma clara e objetiva utilizando meios gráficos. Enfatiza também que para transmitir ideias efetivamente, tanto a forma estética quanto as necessidades funcionais precisam estar equilibradas. Consequentemente, promovendo a compreensão dos dados de forma mais intuitiva.

1.2 Visualização de Dados

A visualização de dados originou-se em diversas áreas do conhecimento, como astronomia e cartografia. No campo da astronomia foram desenvolvidos desenhos do posicionamento das estrelas, galáxias e outros objetos astronômicos, enquanto que na cartografia foram desenvolvidos mapas para auxiliar na navegação marítima e exploração territorial.

No final do século XVII, houve algumas tentativas de mapear informações acerca da economia, saúde e geologia, porém naquele momento havia grande interesse em posições geográficas e nos limites territoriais. Além disso, as preocupações se voltavam para a expansão territorial, navegações marítimas e nas medições físicas, tais como tempo e distância. Neste século também ocorre o crescimento da geometria analítica, as teorias dos erros de medição, e ainda por volta do ano 1650 Pascal inicia as primeiras formalizações da teoria das probabilidades (Friendly, 2009; Young *et al.*, 2006).

Já no século XVIII, inicia-se o crescimento dos gráficos estatísticos e de mapas temáticos, na tentativa de mostrar mais do que apenas mapas com posições geográficas. Isto resulta em novas formas gráficas como, por exemplo, os gráficos de barras e linhas de William Playfair (1759-1823). Apareceram também mapas educacionais na França, e mapeamento de informações acerca da saúde. Inovações tecnológicas dão suporte ao desenvolvimento da visualização de dados, tais como, litografia e impressão a cores. Com o crescimento do volume de dados, as visualizações vão ganhando cada vez mais espaço e novas formas de visualizações

surgem.

Na primeira metade do século XIX, o aumento da importância das informações para o desenvolvimento industrial, social e comercial alavancou o crescimento da visualização de dados. Este crescimento se deve também às inovações obtidas no século passado. Segundo Friendly (2009), os gráficos estatísticos mais conhecidos foram desenvolvidos neste período, como o histograma e o gráfico de série temporal. No âmbito da cartografia foram desenvolvidos mapas temáticos nas mais variadas áreas como econômica, social, médica, *etc.* A segunda metade do século XIX é conhecida como a *era de ouro* da visualização estatística, pois todas as condições para o rápido crescimento da visualização já estavam estabelecidos. Escritórios de análise foram criados em toda Europa, reconhecendo a importância crescente de informações numéricas para os planejamentos sociais e econômicos. Além disso, a teoria estatística desenvolvida por Carl Friedrich Gauss e Pierre Simon Laplace alguns anos atrás forneceu os meios para obter informações de grande volume de dados. Neste mesmo período foi criado por Charles Minard, engenheiro frânces, o famoso gráfico que ilustra a campanha de Napoleão contra a Rússia em 1812, apresentado na Figura 1.1. A largura da trajetória é proporcional ao número de soldados sobreviventes na campanha de guerra. A cor bege indica o caminho realizado pela tropas de Napoleão até Moscou, Rússia. Enquanto que a cor preta indica a trajetória que as tropas fizeram ao voltar da cidade de Moscou. Notamos que há uma diferença considerável entre os números de sobreviventes que saíram da França e número de sobreviventes que voltaram da guerra.

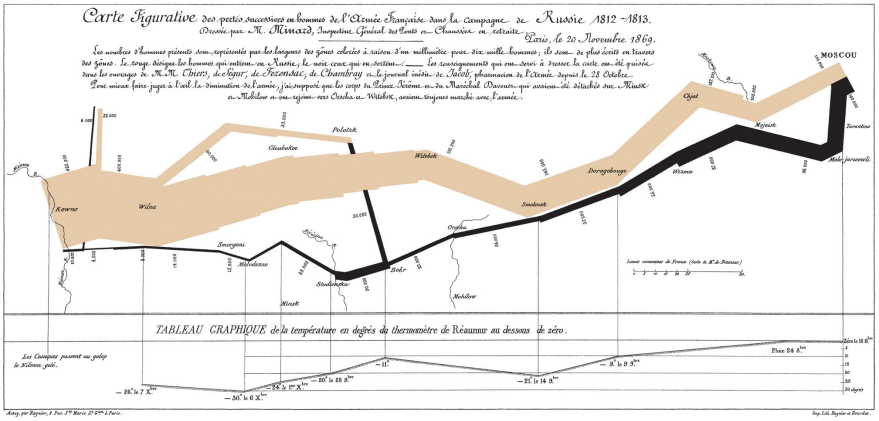


Figura 1.1: *Campanha de Napoleão contra a Rússia em 1812.*

Fonte: Wikipédia, 1 fev. 2013.

Do início do século XX até sua metade, houve poucas contribuições no campo da visu-

alização. Em contrapartida, o desenvolvimento de modelos formais para análise de dados foi substancial. Na segunda metade do século XX, a visualização ganha novamente força nos EUA quando John W. Tukey reconhece a relevância da análise gráfica dos dados e propõe novos padrões de análise, o mais popular é o gráfico *boxplot*, introduzido no seu livro *Exploratory Data Analysis* (Tukey, 1977). Neste livro, Tukey incentiva os estatísticos a examinar visualmente os seus conjuntos de dados, de forma que as hipóteses sobre as causas dos fenômenos observados possam ser sugeridas. Além disso, a análise gráfica pode auxiliar na avaliação das hipóteses em que a inferência estatística será baseada e dar suporte para a seleção de técnicas estatísticas adequadas para o problema (Young *et al.*, 2006). Com o surgimento do computador e o desenvolvimento de softwares e sistemas, as inovações para visualizações de dados surgem a todo momento (Young *et al.*, 2006).

1.3 Justificativa

Na saúde pública, a identificação e quantificação de padrões de ocorrência de doenças fornecem os primeiros passos em direção a uma maior compreensão e, possivelmente, controle de determinada doença. O local de ocorrência de um evento pode fornecer alguma indicação a respeito de porquê esse evento em particular ocorre. Métodos estatísticos espaciais, tais como mapas de incidência de doenças, oferecem um meio para nós utilizarmos as informações de localização para detectar e quantificar os padrões em dados de saúde pública e investigar os potenciais fatores de risco e doenças.

O interesse em visualização de dados georreferenciados no campo da epidemiologia começou com o reconhecimento de mapas como ferramentas úteis para fornecer possíveis "causas" de doença. Um dos estudos mais famosos de epidemiologia espacial foi do Dr. John Snow sobre a epidemia de cólera em Londres, em 1854. Snow foi um médico britânico e é considerado um dos nomes mais importante da bioestatística. Ele acreditava que a cólera era transmitida através da água, porém esta teoria foi recebida com muito desprezo pela sociedade. Portanto, Snow continuou seus estudos e a partir de um mapa de pontos, onde os pontos representavam a localização dos casos de cólera, ficou evidente que em torno da bomba de água da Broad Street estava concentrado um grande número de casos de cólera, como mos-

tra a Figura 1.2. Com o auxílio de técnicas estatísticas e com estudos mais apurados Snow conseguiu encontrar os meios de transmissão da cólera.

Figura 1.2: *Mapa com os casos de cólera identificados por pontos e bombas de água representadas por × na epidemia de Londres (1854).*

Fonte: Wikipédia, 1 fev. 2013.

Outros exemplos de epidemiologia espacial incluem o estudo de raquitismo feito por Palm (1890) , que usou mapas para obter a distribuição geográfica de raquitismo. Palm observou a maior incidência de raquitismo em áreas urbanas industriais onde o clima é frio e úmido. Hoje é sabido que o raquitismo é causado por deficiência de vitamina D, que por sua vez pode ser causada por falta de radiação ultravioleta. Em um estudo um pouco mais recente, Blum (1948) supôs a luz solar como um fator causal para o câncer de pele, mais uma vez baseado principalmente sobre a distribuição geográfica dos casos de doença observados. Estes estudos evidenciam que a localização onde as pessoas vivem tem uma grande importância na identificação de padrões de doenças e que mapas temáticos são amplamente utilizados na identificação destes padrões.

Mapa temático é um instrumento cujo objetivo é expor informações segundo algum tema,

por exemplo, nível socioeconômico estratificado por regiões, taxas de natalidade, mortalidade, dentre outros temas que podem ser visualizados. Além disso, mapa temático é uma forte ferramenta para identificar possíveis causas de um determinado fenômeno físico, social, climático, etc. Porém, a utilização de mapas temáticos pode não ser uma boa solução quando houver a necessidade de representar visualmente mais de um tipo de informação, uma vez que estes tipos de mapas são amplamente apresentados de forma estática, sem que haja interatividade com o analista. Isto dificulta a inserção de mais informações que poderiam auxiliar os analistas de dados. Uma possível solução para este problema é fazer o uso de *geobrowsers* como *Google Earth*, *Google Maps* e *OpenStreetMap*, pois estes permitem uma interatividade maior com os analistas de dados, além de fornecer meios de representar várias informações em único mapa. Outra solução é utilizar outros meios de visualização de dados, como gráficos interativos, capazes de se modificarem instantaneamente com as ações do analista de dados. Alguns pacotes desenvolvidos para o software R estão trabalhando no sentido de facilitar a comunicação entre os analista de dados e os novos *geobrowsers*, bem como novas ferramentas de visualização de dados.

A criação de pacotes com intuito de fornecer ferramentas de visualização de dados tem sido cada vez mais frequente. No trabalho de Nolan (2012), é proposta uma abordagem de criação de interfaces gráficas interativas e animadas usando as linguagens R (Team, 2012) e *Scalable Vector Graphics* (SVG) (Lang, 2012b), uma linguagem com estrutura XML (Lang , 2012c) para descrever interfaces gráficas bidimensionais. Neste trabalho eles desenvolvem o pacote `SVGAnnotation` para o software R, ver Nolan (2012) e Lang (2012b). Outros pacotes em R também estão disponíveis para auxiliar a criação de visualização de dados, como `RGoogleMaps` (Loecher, 2012), que gera mapas estáticos diretamente do Google Maps, sendo possível inserir cores no mapa segundo alguma informação de interesse. Neste mesmo âmbito de visualização georreferenciada, há também os pacotes `RKML` (Lang, 2012a) e `maptools` (Lewin-Koh, 2012) desenvolvidos na plataforma R que permitem desenvolvimento de mapas temáticos e que podem ser visualizados no Google Earth (Google, 2012a) e Google Maps (Google, 2012b), ganhando assim interatividade com o usuário, uma vez que estas ferramentas disponibilizadas gratuitamente pela empresa Google tem essa característica.

Outras ferramentas de visualização de dados surgem na web, como por exemplo *Many Eyes* (IBM, 2012). Este é um sítio público que fornece um sistema de manipulação de dados, que permite aos usuários coletar informações e visualizá-las. O *Google Chart API* é uma ferramenta disponibilizada pela empresa Google para criação de gráficos, como por exemplo, gráfico de barras, linhas, setor, boxplot, entre outros. A principal vantagem desta ferramenta é a facilidade de criação dos gráficos, uma vez que precisa apenas de uma chamada para sua URL enviando os dados do gráfico via *query string*. A chamada realizada retornará a imagem do gráfico. O software livre *Gapminder Desktop* desenvolvido pelo estatístico Hans Rosling, permite que o usuário exiba gráficos estatísticos animados de forma simples e didática.

1.4 Objetivos

Este trabalho de dissertação tem como objetivo a elaboração de um pacote em ambiente R com funcionalidade para a visualização de dados georreferenciados utilizando *APIs* do *Google Maps* e *Google Charts API*. Visa também desenvolver ferramentas para criação de arquivos em linguagem `KML`, que permitirá ao usuário uma interatividade maior com os dados, uma vez que este tipo de arquivo pode ser visualizado dentro do aplicativo *Google Earth*, Google Maps ou na web. Este pacote proposto também terá como funcionalidade a geração de imagens em formato `SVG`, que fornece a opção de criarmos imagens animadas e interativas. Neste trabalho, vamos incorporar a tecnologia *Google Charts* para criação de gráficos dinâmicos e que poderão ser inseridos nos arquivos `KML`, fornecendo ainda mais informação aos mapas gerados.

O foco aplicado deste projeto de pesquisa são os dados médicos. Nossas ferramentas já vem sendo desenvolvidas dentro da Plataforma Arouca, que é um banco de dados que concentra informações sobre os profissionais de saúde que atuam no SUS, reunindo dados sobre o seu histórico educacional e profissional.

Outro objetivo é cálculo o de taxas brutas e bayesianas para cada região. Para mais detalhes acerca do cálculo de taxas bayesianas, ver o trabalho de Marshall (1991). Alguns pacotes desenvolvidos em R já possuem a funcionalidade de cálculo de taxas bayesiana, como, por exemplo, o pacote `spdep`. Pórem, o objetivo deste trabalho é fornecer esta funcionalidade

de forma mais simples e intuitiva.

Como objetivo final esperamos que este pacote funcione com uma interface na web, onde o usuário da Plataforma Arouca não terá a necessidade de entrar com linhas de comandos no software R . Isto vai permitir que um usuário que não esteja familiarizado com ambiente de programação possa gerar visualizações georreferenciadas sem grandes dificuldades.

1.5 Organização do trabalho

Esta dissertação de mestrado é dividida em quatro capítulos. No primeiro capítulo realizamos uma breve revisão histórica das técnicas visualização de dados e descrevemos os objetivos deste trabalho. O capítulo 2 é composto pela metodologia, onde descrevemos as linguagens `KML` , `SVG` e R que irão compor o pacote `aRouca` . Além disso, exemplificamos o funcionamento da tecnologia Google Chart API. No terceiro capítulo apresentamos as funcionalidades do pacote `aRouca` , tais como, mapas do tipo origem–destino, mapas de pontos e de áreas através de exemplos. O quarto e último capítulo descrevemos as contribuições deste trabalho para sociedade e alguns trabalhos futuros.

Capítulo 2

Metodologia

Estatística espacial é uma área da estatística que estuda os fenômenos nos quais as localizações dos eventos são muito importantes, e em alguns casos essenciais para a compreensão do fenômeno. Assim, um conjunto de técnicas estatísticas está disponível para descrição, visualização e análise de dados que possuam posicionamento geográfico. Alguns exemplos de áreas com aplicação de estatística espacial são epidemiologia, agronomia e geologia.

Neste trabalho estamos interessado em métodos para visualização de dados espaciais. Desta forma, destacaremos a utilização e desenvolvimento de mapas temáticos. A seguir, descreveremos alguns mapas temáticos que são utilizados para análise exploratória e apresentação de dados.

Frequentemente dados utilizados em estudos epidemiológicos tem a característica de serem eventos pontuais. Ou seja, estão associados com pontos geográficos específicos. Por exemplo, as localizações exatas das residências dos indivíduos com Dengue. Assim, um mapa de pontos é útil caso o objetivo seja visualizar a distribuição espacial dos eventos pontuais. Mapas de pontos são frequentemente utilizados para monitorar a propagação de doenças infecciosas. Além disso, eles são úteis na identificação de potenciais fontes de epidemias. Outro mapa comumente utilizado para apresentação de eventos pontuais é o mapa de pontos proporcionais. Este mapa é utilizado quando estamos interessados em apresentar a localização espacial do evento e, associado ao evento, temos alguma variável de interesse. Esta variável é representada pelo tamanho do ponto. Ou seja, quanto maior o ponto, maior é o valor da variável.

Outro tipo de mapa temático muito comum é o mapa de área. Este tipo de mapa é comumente utilizado para representar dados espaciais cuja localização está associada a áreas delimitadas por polígonos. Frequentemente os eventos cuja localização exata não está disponível são agregados em áreas, estas áreas por sua vez podem ser representadas por municípios, setores censitários, unidades federativas, *etc.* Para construção deste mapa, o atributo (variável) que será representado no mapa é classificado em diferentes categorias de interesse do analista de dados. Em seguida, diferentes cores são utilizadas para representar cada categoria do atributo. Ou seja, cada área é classificada em uma das categorias, e colorida segundo a classificação do atributo.

Mapas temáticos podem ser criados em diversos Sistemas de Informação Geográfica, ou mesmo em software estatísticos. Neste trabalho iremos trabalhar com as ferramentas disponíveis em ambiente R (Team, 2012), destacando os pacotes de análise dados georreferenciados, como `maptools` e `spdep`. Estes pacotes serão úteis na construção de mapas de pontos e de áreas. Além destes pacotes também utilizaremos os recursos das linguagens `KML` e `SVG` para o desenvolvimento dos mapas dinâmicos e interativos. Utilizaremos também os pacotes citados anteriormente para o desenvolvimento das visualizações georreferenciadas.

Quanto as funcionalidades disponíveis pela empresa Google, como *Google Charts API*, para criação de gráficos, e Google Earth para visualização de dados georreferenciados, serão também incorporadas neste trabalho. Desta forma, será desenvolvida a integração do software R e as funcionalidade da Google.

2.1 Software R

O R é uma linguagem e um ambiente computacional para desenvolvimento de técnicas e análises estatísticas, e também de gráficos. Foi desenvolvida originalmente por Ross Ihaka e Robert Gentleman no departamento de Estatística da universidade de Auckland, Nova Zelândia e é similar ao pacote comercial S-Plus. Entretanto, é um projeto baseado no conceito de software livre e pode ser utilizado sem que haja nenhum custo de licença. O código fonte do R está disponível sob a licença GNU GPL.

Diferentemente da maioria dos softwares estatístico, o R é acessado via linha de comando,

porém existem várias interfaces gráficas (GUI) para o R , incluindo JGR, Rcmdr e, mais recentemente RStudio. Na sua forma usual, o R possui três ambientes: *prompt* de comando, área gráfica e o ambiente para desenvolvimento de script (*debugger*).

O R possui uma ampla variedade de técnicas estatísticas implementadas, tais como modelagem linear e não-linear, análises de séries temporais, testes estatísticos clássicos, dentre outras. Além disso, funciona em diversos sistemas operacionais, como: Windows, Linux e MacOS.

Outro aspecto interessante do R é sua enorme capacidade de visualização de dados. O desenvolvimento de gráficos a partir do R permite a manipulação dos mais variados parâmetros gráfico. Desta forma, o usuário tem o controle total dos gráficos que deseja construir, proporcionando um ambiente bastante flexível para desenvolvimento e apresentação de dados. Outra característica importante é a grande coleção de ferramentas para análise e manipulação de dados.

Algumas funções mais simples ou mais populares já estão disponíveis no R , sem que haja a necessidade de instalação de nenhum módulo adicional. Porém, quando houver a necessidade de utilizar alguma técnica mais específica e esta não estiver disponível na base do R , podemos então instalar um pacote auxiliar que possua a técnica implementada. Estes pacotes são bibliotecas compostas por um conjunto de funções específicas e podem ser baixados através do *Comprehensive R Archive Network* (CRAN) ou por outro repositório. Com a característica de ser um software livre, os usuários do R podem contribuir no seu desenvolvimento através de novos pacotes ou no próprio código fonte do R . Desta forma o R é altamente expansível com o uso de pacotes. Para mais detalhes acerca da construção de pacotes do R , ver Hornik (2012).

A integração do R com outras linguagens de programação é amplamente utilizada para melhorar o desempenho de alguns procedimentos. Entre elas estão C, C++ e FORTRAN. Além de interagir com estas linguagens, o R também comunica com outros softwares, tais como Microsoft Excel, OpenBugs e Statistical Analysis System (SAS). Outras integrações também são realizadas com o intuito de fornecer novas ferramentas de visualização de dados. Por exemplo, a integração com a linguagem `KML` , que foi desenvolvida pela Google para exibir informações georreferenciadas através do Google Maps e Google Earth. Estas integrações

contribuem amplamente na potencialidade do R como software de análise e visualização de dados.

2.1.1 Classe de dados espaciais

Neste trabalho utilizaremos o formato *shapefile* como padrão para os mapas. Este formato foi desenvolvido pela empresa americana ESRI (Environmental Systems Research Institute), especializada na área de informações geográficas e é amplamente utilizado pelos Sistemas de Informações Geográficas (SIG's). O formato *shapefile* é composto por um conjunto de pelo menos três arquivos: um arquivo contendo as coordenadas geográficas dos polígonos, pontos ou linhas (extensão .shp), outro com atributos (extensão .dbf), por último um arquivo com os índices (extensão .shx). Outro arquivo que pode compor o formato *shapefile* é o arquivo que indica a projeção cartográfica dos polígonos, linhas ou pontos, sua extensão é dada por .prj.

Dados espaciais são compostos principalmente por duas estruturas básicas: pela informação georreferenciada (geometrias) e por seus atributos. Ambas estruturas muitas das vezes não podem ser representadas de forma simples, como uma matriz de dados. Por exemplo, imaginamos que temos o registro do número de óbitos por acidente de carro e a população de cada município (atributos), sendo os municípios representados por polígonos (geometrias). Neste caso, a combinação da estrutura dos polígonos e o seus atributos não pode ser representada por uma estrutura simples de dados como uma matriz. Desta forma, fica exemplificado que dados espaciais devem ser representados por estruturas que acomodem bem as informações georreferenciadas e os seus atributos (Ribeiro, 2011).

Além disso, dados espaciais são comumente divididos em três subáreas que dependem do formato específico dos dados e dos modelos a serem utilizados. Estas subáreas são dadas pela variação espacial discreta, contínua ou de processos pontuais (Ribeiro, 2011).

Portanto, com o intuito de desenvolver uma estrutura que contemplasse os diferentes formatos de dados espaciais, Pebesma e Bivand (2005) desenvolveram o pacote `sp`. O pacote `sp` fornece classes e métodos para lidar com dados espaciais no R . As estruturas de dados implementadas no pacote incluem pontos, linhas, polígonos e grades, sendo cada uma delas com ou sem dados de atributo. Para representação dos objetos espaciais em R , o pacote `sp`

utiliza classes e objetos do tipo S4 (Bivand *et al.*, 2008).

Os dados contidos em um objeto da classe S4 são definidos por *slots*, sendo que cada *slot* de um objeto é um componente da classe S4. Assim como os elementos de uma lista, estes podem ser extraídos usando a função `slot()` ou o operador "@" (R Development Core Team , 2012). No entanto, os *slots* se diferenciam dos componentes de uma lista de uma maneira importante: os *slots* não podem ser consultados por posição como nas listas, apenas referenciados pelos nomes atribuídos a eles. Abaixo segue um exemplo em R ilustrando objetos da classe S4.

```
1 > library(maptools) #carregando o pacote maptools, consquentemente o pacote sp.
2 > sids <- readShapeSpatial(system.file("shapes/sids.shp", package="maptools")[1], IDvar="FIPSNO",
      proj4string=CRS("+proj=longlat +ellps=clrk66"))
3 > slotNames(sids) #consultado os nomes dos slots.
4 [1] "data"        "polygons"    "plotOrder"   "bbox"        "proj4string"
5 > slot(sids, "data") #consultando o slot "data" via função slot()
6 > sids@data #consultando o slot "data" via operador "@"
7 > sids@polygons
8 > slotNames(sids@polygons[[1]])
9 [1] "Polygons"  "plotOrder" "labpt"     "ID"        "area"
```

Observamos que na linha 8 do código acima estamos consultando o *slot "polygons"*, que por sua vez é da classe `list`, porém a primeira posição desta lista é um objeto da classe S4. Portanto, podemos utilizar a função `slotNames` para consultarmos quais *slots* compõem este objeto. Este pequeno trecho de código R ilustra que a classe definida pelo pacote `sp` para representar dados espacias é eficiente, pois consegue combinar bem a estrutura espacial e seus atributos.

Desta forma, a maioria dos pacotes desenvolvidos para análise de dados espaciais utiliza o formato de dados definido pelo pacote `sp`, ou então possuem formas de converter para o formato específico do `sp`.

Para leitura de mapas em R existem várias opções de pacotes e funções. Neste trabalho utilizaremos o pacote `maptools` (Lewin-Koh, 2012) para leitura de arquivos *shapefiles*, pois este utiliza as classes de dados espaciais definidas pelo pacote `sp` (Bivand *et al.*, 2008).

2.2 Linguagem XML

A XML - **eXtensible Markup Language** é uma linguagem de marcação que define um conjunto de regras para a codificação de documentos em um formato que seja simples e legível tanto para as pessoas quanto para os computadores. Os objetivos do XML ressaltam a simplicidade, generalidade, e usabilidade através da internet. Além disso, é um formato de dados que suporta Unicode, permitindo que a maior parte da informação codificada em linguagem humana possa ser transmitida (Lang, 2012c).

As componentes de um documento XML são basicamente os marcadores (tags) e o conteúdo presente entre as tags. Sequências que constituem um marcador inciam com o caractere < e finalizam com >. Sequências de caracteres que não são marcadores formam os conteúdos. Os marcadores são caracterizados em três grupos:

- ***start-tags***: `<section>`
- ***end-tags***: `</section>`
- ***empty-elements tags***: `< line-break />`

Documentos XML são criados segundo uma hierarquia de elementos a partir de uma única tag raiz. Além disso, XML é *case sensitive*. Portanto, letras maiúsculas diferem de minúsculas. As tags não são pré-determinadas como em outras linguagens de marcação como HTML. A estrutura hierárquica dos marcadores pode ser ilustrada na Figura 3.13.

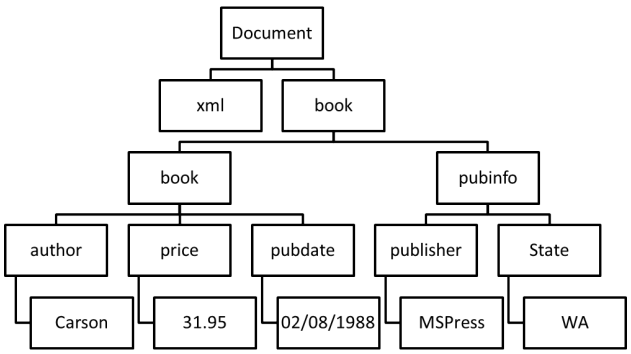


Figura 2.1: *Estrutura hierarquica da linguagem XML.*

XML permite ao programador criar suas próprias tags, ou seja, a partir das tags criadas pode-se determinar uma outra "linguagem" seguindo as regras destas tags criadas. Esta

característica do XML faz com que esta linguagem seja amplamente utilizada. Exemplos de linguagens que utilizam a estrutura XML são SVG e KML.

2.2.1 Linguagem SVG

SVG é uma abreviatura para *Scalable Vector Graphics*. Esta linguagem descreve de forma vetorial desenhos e gráficos bidimensionais em XML, podendo ser criado tanto na forma estática, quanto dinâmica ou animada. A principal característica dos gráficos vetoriais é que estes não perdem qualidade uma vez que são ampliados, ao contrário de gráficos raster, como ilustrado na Figura 2.2. Isto ocorre pelo fato do SVG ser um sistema baseado em vetores que descreve uma imagem como uma série de formas geométricas Lang (2012b). Documentos do tipo SVG incluem os comandos para desenhar formas em um conjuntos específico de coordenadas, implicando em uma imagem de alta qualidade. Outra característica de documentos SVG é o tamanho dos arquivos, pois estes são compostos apenas por textos, fazendo com que os arquivos sejam extremamente pequenos quando comparados aos arquivos de imagem raster. Outro atrativo para esta linguagem comparado a outros formatos vetoriais é o fato do SVG ser um formato aberto, ou seja, não é propriedade de nenhuma empresa. Esta linguagem foi criada pela *World Wide Web Consortium*, responsável pelo desenvolvimento de outros padrões, como o HTML e o XHTML. Ressalta-se também que a linguagem SVG é suportada por todos navegadores *Web* de forma nativa ou através de alguma biblioteca *JavaScript*.

A linguagem SVG permite três tipos de objetos gráficos:

- formas gráficas vetoriais, por exemplo, linhas, curvas, polígonos;
- imagens (vetorial ou raster);
- texto.

O armazenamento de imagens do tipo SVG pode ser gravadas com a compressão do tipo *zip*, sem ocasionar nenhuma perda de informação. Porém após a compressão a extensão do arquivo deve ser SVGZ.

2.2.1.1 Descrição do SVG

A tradução da sigla SVG pode ser feita como Gráficos Vetoriais Escaláveis:

Gráfico : As linguagens do tipo XML tem certa deficiência gráfica, permitindo apenas informação do tipo texto, ou carregando alguma imagem do tipo png ou jpeg. Desta forma se fez necessário a criação de uma linguagem com capacidade gráfica e que pudesse ser incorporada em algum código do tipo XML.

Vetoriais : Os dois tipos de gráficos mais comuns são raster e vetoriais. Gráficos raster são baseados em *pixels* para representar uma imagem, enquanto que os vetoriais são baseados em linhas, pontos, curvas e polígonos (formas geométricas). Uma vez que os gráficos vetoriais armazenam a equação dos gráficos e não um mapa de pixels como imagens raster, estes não sofrem com o efeito de ”despixalização”quando são ampliados. Outra vantagem é o fato de que imagens maiores ocupam o mesmo espaço de que imagens menores, sem que haja perda de qualidade. Eles permitem também a criação de animações.

Escaláveis : Escalonamento é o efeito de aumentar e diminuir de forma uniforme uma imagem. Gráficos vetoriais permitem *zoom* rápido e eficiente sem perda de qualidade.

O conjunto destes três conceitos define as capacidades do SVG e justifica a sua criação como a tecnologia de gráficos para a *World Wide Web*.

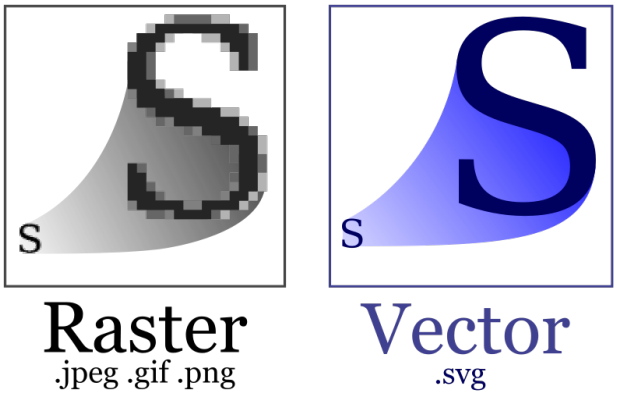


Figura 2.2: *Imagem comparativa entre uma imagem raster e vetorial.*

2.2.1.2 Gramática SVG

Documentos SVG permitem várias formas, sendo algumas: retângulos, círculos, linhas, elipses, polígonos e caminhos. Para criarmos estes objetos é preciso ter um mínimo conhecimento da linguagem XML ou alguma outra linguagem de marcação (tag).

- `<rect>` - Retângulo
- `<circle>` - Círculo
- `<ellipse>` - Elipse
- `<line>` - Linha
- `<polygon>` - Polígono

Objeto Retângulo

A tag `<rect>` é usada para criar a forma retângulo e suas características, como cor, tamanho, entre outras. A sintaxe é a seguinte:

```
<rect x="" y="" width="" height="" rx="" ry="" style="" />
```

Os atributos `x` e `y` são utilizados para identificar o canto superior esquerdo do retângulo, enquanto que os atributos `width` e `height` são as dimensões do mesmo. Os atributos `rx` e `ry` são elementos opcionais e indicam o centro de um arco que cria o efeito de arredondamento dos cantos do retângulo. O último atributo, `style`, é utilizado quando aplicamos alguma propriedade de folhas de estilos.

```
1  <svg xmlns="http://www.w3.org/2000/svg" width="
       100%" height="100%">
2
3       <rect x="50" y="50" width="90" height="90"
            rx="30" ry="30"
4             stroke="green" fill-opacity="0"
                  stroke-width="1"
5             stroke-dasharray="5 5"/>
6
7       <rect x="70" y="40" width="90" height='90'
8             stroke="green" fill-opacity="0"
                  stroke-width="1"/>
9
10    <rect x="190" y="50" width="90" height='90'
11          stroke="green" fill="blue" fill-opacity
                ="0.2"
12            stroke-width="20" stroke-opacity="0.2
                  "/>
13 </svg>
```

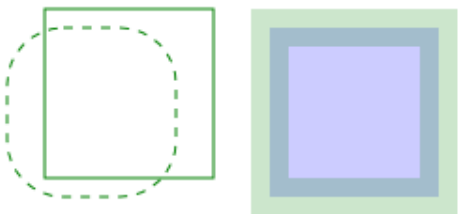


Figura 2.3: *Exemplo da linguagem* `SVG` *na criação e objetos do tipo retângulo.*

Objeto Círculo

A tag `<circle>` é usada para criar forma em círculo.

Sintaxe:

```
<circle cx="" cy="" r="" style="" />
```

- `cx`: coordenada do centro do círculo referente ao eixo x.
- `cy`: coordenada do centro do círculo referente ao eixo y.
- `r`: raio

```
1  <svg xmlns="http://www.w3.org/2000/svg"
2       width="100%" height="100%">
3       <circle cx="200" cy="200" r="50"
4             stroke="green" fill-opacity="0"
                  stroke-width="1"
5             stroke-dasharray="5 5"/>
6       <circle cx="300" cy="300" r="75"
7             stroke="green" fill-opacity="0"
                  stroke-width="1"/>
8     <circle cx="400" cy="400" r="100"
9           stroke="green" fill="blue" fill-opacity
                ="0.2"
10        stroke-width="20" stroke-opacity="0.2"/>
11 </svg>
```

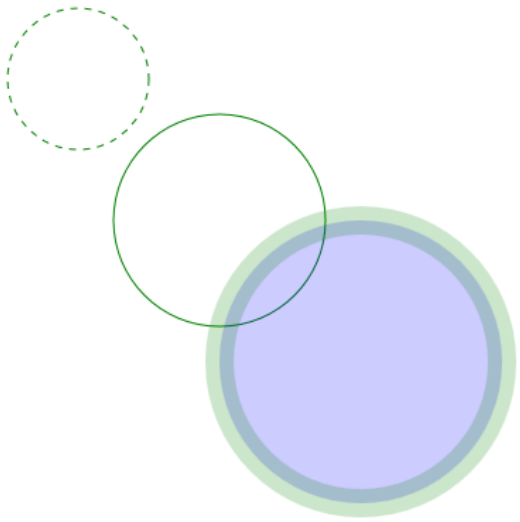


Figura 2.4: *Exemplo da linguagem* `SVG` *na criação e objetos do tipo circulo.*

Objeto Linha

A tag `<line>` é usada para criar forma de uma linha.

```
<line x1="" y1="" x2="" y2="" style="" />
```

Os parâmetros `x1, y1, x2` e `y2` são usados para indicar o ponto de início e o ponto de fim da linha. O último atributo, `style`, é semelhante ao retângulo e círculo.

Existem outros tipos de objetos em `SVG`, porém estaremos interessados neste primeiro momento apenas neste três objetos.

2.2.1.3 Exemplo de SVG

Documentos SVG são iniciados com a tag <svg> e finalizado com </sgv>, ou seja, a estrutura é a mesma que uma linguagem do tipo XML. SVG fornece uma estrutura para organizar o documento de forma que não haja repetições dentro do documentos. Esta estrutura é feita de forma a agrupar definições de formas, estilos, figuras, dentre outros objetos que o programador julgar necessário. Estes agrupamentos permitem reutilizar estes componentes, tornando o código mais eficiente e menor. Agrupamentos são construídos com a tag <g>, sendo que cada grupo possui um identificador (id), permitindo ao longo do documento do SVG realizar requisições destes grupos. Abaixo segue um exemplo de um documento SVG estruturado (Rocha, 2012).

```
1  <svg xmlns:xlink="http://www.w3.org/1999/xlink"
       xmlns="http://www.w3.org/2000/svg"
2    width="1050" height="600" viewBox="0 0 1050
         600">
3    <defs>
4      <rect id="preto" x="0" y="0" width="20"
           height="20" fill="rgb(0,0,0)" />
5      <rect id="branco" x="0" y="0" width="20"
           height="20" fill="rgb(255,225,250)" />
6      <g id="quatroCasas">
7        <use xlink:href="#preto"/>
8        <use xlink:href="#branco" transform="
             translate(20)"/>
9        <use xlink:href="#branco" transform="
             translate(0,20)"/>
10       <use xlink:href="#preto" transform="
             translate(20,20)"/>
11     </g>
12     <g id="fileiraDupla">
13       <use xlink:href="#quatroCasas"/>
14       <use xlink:href="#quatroCasas" transform="
             translate(40)"/>
15       <use xlink:href="#quatroCasas" transform="
             translate(80)"/>
16       <use xlink:href="#quatroCasas" transform="
             translate(120)"/>
17     </g>
18     <g id="tabuleiro">
19       <use xlink:href="#fileiraDupla"/>
20       <use xlink:href="#fileiraDupla" transform=
             "translate(0,40)"/>
21       <use xlink:href="#fileiraDupla" transform=
             "translate(0,80)"/>
22       <use xlink:href="#fileiraDupla" transform=
             "translate(0,120)"/>
23     </g>
24   </defs>
25   <use xlink:href="#tabuleiro"/>
26 </svg>
```

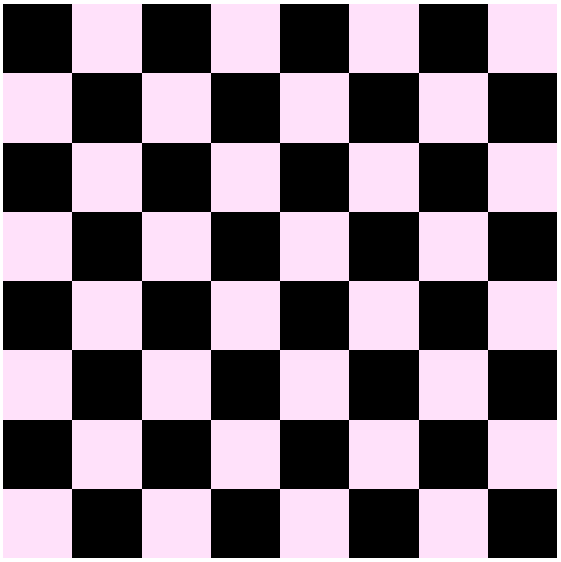


Figura 2.5: *Exemplo da linguagem* SVG *na criação e objetos do tipo circulo.*

2.2.2 Linguagem KML

Keyhole Markup Language (KML) é uma linguagem baseada na estrutura XML , ou seja, utiliza-se tags com elementos e atributos aninhados, a fim de expressar registros geográficos e visualização de conteúdos existentes nessa linguagem como mapas em 2D e navegadores terrestre em 3D, como Google Maps e Google Earth. Como o KML é baseado em XML , algumas características do XML são herdadas, como por exemplo, as tags diferenciam maiúsculas e minúsculas e devem aparecer exatamente como listadas em Google (2012c). Algumas tags são opcionais e dependendo do contexto a ordem destas irá influenciar na estrutura do arquivo KML .

O arquivo contendo o código KML poderá ser visualizado em vários aplicativos, incluindo Google Earth, Google Maps, Google Maps para celular, NASA WorldWind, ESRI ArcGIS Explorer, Adobe PhotoShop, AutoCAD e Yahoo! Pipes. Além disso, estes arquivos podem ser incorporados em algum sítio na web.

2.2.2.1 Gramática KML

Os documentos de KML mais simples podem ser criados diretamente no Google Earth, isto é, não há necessidade de editar ou criar um arquivo KML em um editor de texto. Marcadores, superposições de solo, caminhos e polígonos podem ser criados diretamente no aplicativo Google Earth.

Marcadores

Marcador é um recurso amplamente utilizado no Google Earth, uma vez que este objeto proporciona marcar uma posição na superfície da Terra. O marcador mais simples inclui somente um elemento <Point> (ponto), que especifica o local do marcador. Pode-se especificar um nome e um ícone personalizado para o marcador, assim como pode acrescentar outros elementos geométricos a ele.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <kml xmlns="http://www.opengis.net/kml/2.2">
3   <Placemark>
4     <name>Simple placemark</name>
```

```
5     <description>Attached to the ground. Intelligently places itself
6        at the height of the underlying terrain.</description>
7     <Point>
8       <coordinates>-122.0822035425683,37.42228990140251,0</coordinates>
9     </Point>
10   </Placemark>
11 </kml>
```

A estrutura do código acima é dada da seguinte forma:

- Cabeçalho `XML` é primeira linha de todo arquivo KML. Não se usa espaço nem outros caracteres antes dessa linha.
- Declaração de `namespace` através do atributo `xmlns`, pois esta define a versão do `KML`. Esta declaração é a segunda linha de todo arquivo `KML`.
- Um objeto de marcador que contém estes elementos:
  - `name`: usado como rótulo do marcador;
  - `description`: conteúdo que será mostrado pelo "balão" anexado ao marcador, este parâmetro permite inserir links, códigos HTML, vídeos;
  - `Point`: especifica a posição do marcador na superfície da Terra (longitude, latitude e altitude opcional).

Polígonos

Um polígono é definido pelos limites externos e internos. Os limites são definidos por `LinearRings`. Polígonos usam `<PolyStyle>` para cor, modo de cor e preenchimento. Este recurso pode ser utilizado na criação de mapas temáticos, como pode ser visualizado na Figura 2.6

Abaixo segue um breve exemplo de um arquivo `KML` contendo a estrutura para criar um polígono. No exemplo da Figura 2.6 o documento fica muito extenso, pois estamos criando um polígono para cada município de Minas Gerais, resultando em 853 polígonos, lembrando que para cada polígono temos que inserir as latitudes e longitudes de cada ponto do polígono

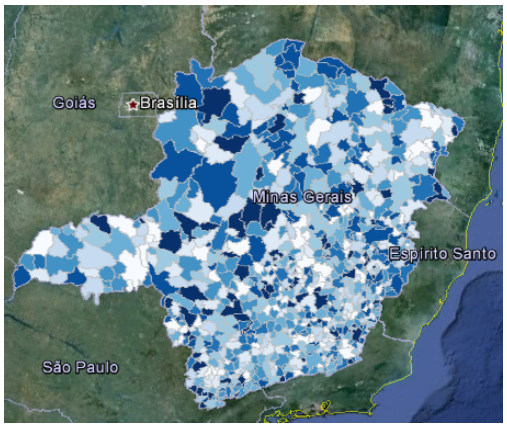


Figura 2.6: *Exemplo de mapa temático utilizado o recurso de polígonos da linguagem* KML

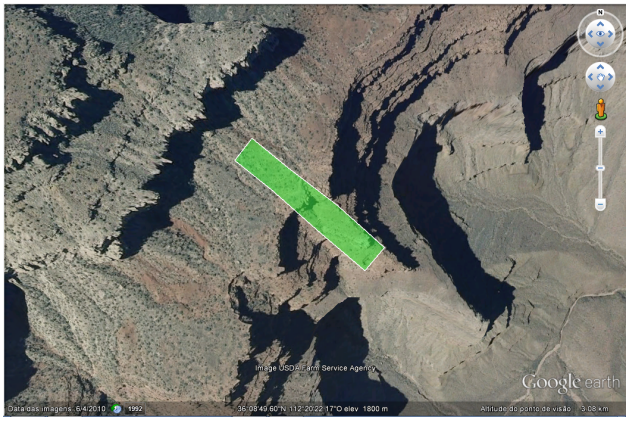


Figura 2.7: *Exemplo utilizado o recurso* Polygon *da linguagem* KML *para representar um polígono simples.*

que define os seus limites. Desta forma fica clara a necessidade de usarmos ferramentas computacionais na criação destes tipo de mapas temáticos. Este exemplo é ilustrado pela Figura 2.7 e possui uma estrutura bem simples, com intuito apenas de ilustrar alguns recursos da linguagem KML .

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <kml xmlns="http://www.opengis.net/kml/2.2">
3    <Document>
4      <name>KmlFile</name>
5      <Style id="transGreenPoly">
6        <LineStyle>
7          <width>1.5</width>
8        </LineStyle>
9        <PolyStyle>
10         <color>7d00ff00</color>
11       </PolyStyle>
12     </Style>
13     <Placemark>
14       <name>Relative</name>
15       <visibility>1</visibility>
16       <styleUrl>#transGreenPoly</styleUrl>
17       <Polygon>
18         <tessellate>1</tessellate>
19         <altitudeMode>relativeToGround</altitudeMode>
20         <outerBoundaryIs>
21           <LinearRing>
22             <coordinates>
23               -112.3349463145932,36.14988705767721,100
```

```
24                -112.3354019540677,36.14941108398372,100
25                -112.3344428289146,36.14878490381308,100
26                -112.3331289492913,36.14780840132443,100
27                -112.3317019516947,36.14680755678357,100
28                -112.331131440106,36.1474173426228,100
29                -112.332616324338,36.14845453364654,100
30                -112.3339876620524,36.14926570522069,100
31                -112.3349463145932,36.14988705767721,100
32              </coordinates>
33            </LinearRing>
34          </outerBoundaryIs>
35        </Polygon>
36      </Placemark>
37    </Document>
38  </kml>
```

Documentos do tipo KML permitem ainda inserir alguma imagem estática, este recurso é mais utilizado quando há necessidade de inserir uma legenda no mapa, pois não estaremos interessados que a legenda sofra alguma alteração quando utilizado a ferramenta de zoom ou qualquer outra que altere o dimensionamento da imagem.

Além destes recursos, a linguagem KML suporta componentes temporal a partir da tag <timestamp>, ou seja, se estivermos com interesse de visualizar alguma informação georreferenciada ao longo de um período de tempo, então podemos também fazer uso do KML .

2.3 Google Chart API

O *Google Chart API* é uma ferramenta disponibilizada pela empresa Google para criação de gráficos, como por exemplo , gráfico de barras, linhas, setor, boxplot, entre outros. A principal vantagem desta ferramenta é a facilidade de criação dos gráficos, uma vez que precisa apenas de uma chamada para sua URL enviando os dados do gráfico via *query string*. A chamada realizada retornará a imagem do gráfico. Neste trabalho estaremos interessados no gráfico de barras, com intuito apenas de exemplificar a ferramenta *Google Chart API*.

A sigla *API* do inglês, *Application Programming Interface* (ou Interface de Programação de Aplicativos) é um conjunto de rotinas e padrões estabelecidos por um software para

serem utilizadas por aplicativos que não tem a intenção de entrar em detalhes acerca da implementação do software, mas apenas a utilização do serviço. No contexto de web, uma API pode ser resumida em conjunto definido de mensagens de requisição e respostas do tipo HTTP.

O modo mais simples desta ferramenta para desenvolver gráficos e disponibilizá-los em uma página web é utilizar o assistente *chart wizard*, disponível em http://code.google.com/intl/pt-BR/apis/chart/image/docs/chart_wizard.html (acessada em 1 de agosto de 2012). Este assistente gráfico fornece um menu na tela apresentando os recursos que podem ser adicionados ao gráfico, sendo selecionados através do mouse ou inserção de caracteres.

A vantagem desta ferramenta é que todas as informações acerca do gráfico que se deseja, como os dados a serem visualizados, tamanho, cores e rótulos fazem parte da URL. Alguns parâmetros gráficos são obrigatórios na URL para que o gráfico seja reproduzível, estes são: o tipo do gráfico (`cht`), os dados (`chd`) e tamanho (`chs`). Já os parâmetros adicionais dependerão de qual tipo de gráfico o indivíduo tem interesse. A lista de parâmetros gráficos pode ser encontrada em http://code.google.com/intl/pt-BR/apis/chart/image/docs/chart_params.html (acessada em 1 de agosto de 2012). A URL pode ser digitada diretamente no navegador, ou inseri-la em uma tag `<img>` em uma página web. Todas as URLs tem o seguinte formato:

```
1 https://chart.googleapis.com/chart?cht=<chart_type>&chd=<chart_data>&chs=<chart_size>&...
      additional_parameters...
```

A elaboração das URLs, devem primeiramente iniciar com a string

```
1 https://chart.googleapis.com/chart?
```

e seguidos dos parâmetros obrigatórios, descritos anteriormente e dos parâmetros adicionais caso necessário. Ressaltando que os parâmetros são inseridos na string da URL com o seu respectivo nome e o valor (nome=valor), e são separados pelo sinal &.

2.3.1 Exemplo: Gráfico de Barras

O exemplo mais simples para o gráfico de barras é adicionando apenas os parâmetros gráficos obrigatórios na URL, ou seja,

```
1 http://chart.apis.google.com/chart?cht=bvs&chd=t:10,20,30,20,10&chs=170x100
```

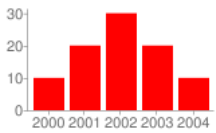
Os parâmetros utilizados foram:

- `cht=bvs`: tipo de gráfico solicitado (`bvs` refere ao gráfico de barras);
- `chd=t:10,20,30,20,10`: os dados são enviado em formato texto (t) separados por vírgula;
- `chs=170x100`: o tamanho da imagem é 170 × 100 pixels.

Podemos adicionar parâmetros referentes à cor do gráfico e aos rótulos dos eixos. Para a especificação da cor é usada uma string de seis caracteres hexadecimais obrigatórios, e mais dois caracteres opcionais, referentes a transparência, onde `00` é completamente transparente e `FF` completamente opaco, sendo o último considerado como padrão para o nível de transparência. Desta forma, o formato de cor é da forma `RRGGBB[AA]`, onde `RGB` são referentes as cores vermelho (red), verde (green), e azul (blue). Atualizando a URL anterior temos o sequinte:

```
1 http://chart.apis.google.com/chart?cht=bvs&chd=t:10,20,30,20,10&chs=170x100&chxt=x,y&chxr
    =0,2000,2004&chco=FF000055&chds=a
```

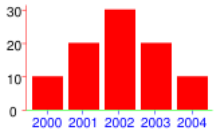
- `chxt=x,y`: visualizar os rótulos dos eixos `x` e `y`;
- `chxr=0,2000,2004`: renomear os rotulos do eixo `x` (0), iniciando em 2000 e finalizando em 2004, caso queira modificar o eixo `y` (1) também, basta fazermos `chxr=0,2000,2004|1,0,3`;
- `chco=FF000055`: atribuindo cor vermelha às barras do gráfico e inserindo transparência;
- `chds=a`: escala do eixo `y` automática.



Podemos modificar também o estilo dos eixos, como por exemplo, a cor e o tamanho da fonte dos rótulos, bem como a cor dos ticks. O parâmetro a ser adicionado na URL, é `chxs`.

```
1 http://chart.apis.google.com/chart?cht=bvs&chd=t:10,20,30,20,10&chs=170x100&chxt=x,y&chxr
    =0,2000,2004&chco=FF0000&chds=a&chxs=0,0000FF,10,0,lt,0000FF,00FF00|1,000000,10,0,lt,FF0000,
    FF0000
```

- `chxs=0,0000FF,10,0,lt,0000FF,00FF00|1,00FF00,10,0,lt,FF0000,FF0000`: o primeiro atributo é o índice do eixo a ser modificado, são eles 0,1,2,3, iniciando no eixo que é comumente conhecido por eixo `x` e em sentido anti-horário são os demais. Na sequência temos a cor (`0000FF`) do rótulo, o tamanho (`10`) da fonte em pixels, alinhamento (`0`) dos rótulos junto aos ticks, neste exemplo estão centralizados nos ticks. O próximo atributo (`lt`) refere-se a desenhar as linhas dos eixos e os ticks, neste exemplo estamos adicionando ambos, porém pode-se optar em adicionar apenas um dos dois (`l` ou `t`) ou nenhum (_). Os dois últimos atributos são referentes a cor do eixo e do tick respectivamente. Para os demais eixos o procedimento é similar, necessitando apenas do separador `|` para determinar a configuração de cada eixo.



Quanto às barras, podemos modificar a largura e o espaçamento entre as mesmas. Desta forma, basta adicionar mais um parâmetro na string da URL como os dois atributos de interesse, resultando na sequinte string:

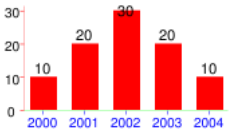
```
1 http://chart.apis.google.com/chart?cht=bvs&chd=t:10,20,30,20,10&chs=170x100&chxt=x,y&chxr
    =0,2000,2004&chco=FF0000&chds=a&chxs=0,0000FF,10,0,lt,0000FF,00FF00|1,000000,10,0,lt,FF0000,
    FF0000&chbh=20,11
```

- `chbh=20,11`: o primeiro atributo do parâmetro `chbh` refere-se a largura das barras em pixels, e o segundo atributo é referente ao espaçamento entre as barras, também em pixels.

Caso estejamos interessados em exibir os valores das respectivas barras, então adicionamos o parâmetro `chm`, tendo como resultando a URL:

```
1 http://chart.apis.google.com/chart?cht=bvs&chd=t:10,20,30,20,10&chs=170x100&chxt=x,y&chxr
    =0,2000,2004&chco=FF0000&chds=a&chxs=0,0000FF,10,0,lt,0000FF,00FF00|1,000000,10,0,lt,FF0000,
    FF0000&chbh=20,11&chm=N,000000,0,-1,11
```

- `chm=N,000000,0,-1,11`: O primeiro atributo é a respeito do tipo de marcador iremos usar nas barras, que para este exemplo é o próprio valor da barra (`N`), já o segundo `000000` é a cor dos rótulos das barras, quanto ao terceiro atributo refere-se ao índice (`0`) dos dados, no caso de termos mais de um grupo no gráfico. O atributo `-1` significa que será exibido os rótulos de todas as barras, caso o atributo fosse `-2`, então seria exibido apenas as barras nas posições multiplas de dois, incluindo a primeira barra. O útimo atributo deste exemplo é o tamanho da fonte do rótulo.



Desta forma exemplificamos uma parcela do potencial desta ferramenta, destacando a quantidade de opções para customizar os gráficos, e as diversas combinações de parâmetros, dando ao usuário uma flexibilidade muito grande. Podemos destacar também a facilidade de integrar essa ferramenta a um banco de dados para criação de gráficos dinâmicos. Vale ressaltar que o número de acessos às URLs é ilimitado, salvo quando o Google julgar abusivo a frequência das requisições.

Capítulo 3

Pacote `aRouca`

3.1 Plataforma Arouca

A Plataforma Arouca é um ambiente que agrega informações sobre os profissionais de saúde que atuam no Sistema Único de Saúde (SUS), reunindo dados sobre histórico educacional e profissional. Os dados que compõem a plataforma se encontram dispersos em várias fontes, tais como Ministério da Saúde/Cadastro Nacional de Estabelecimentos de Saúde - CNES, secretarias estaduais e municipais, Comissão Nacional de Residência Médica, sistemas de gestão acadêmica de universidades, escolas de saúde, entre outras.

Estas informações disponíveis na internet através da Plataforma Arouca servirá de apoio aos gestores nos processos de planejamento, monitoramento e avaliação das ações educacionais, possibilitando um maior conhecimento do perfil profissional de seus servidores, das demandas de capacitação e das ofertas educacionais existentes no seu território. Além disso, o monitoramento e avaliação das ações de educação promovidas pela instituição, mapeamento de profissionais com competências específicas comprovadas em temas ou conteúdos de interesse da gestão e rastreamento das trajetórias profissionais de egressos de cursos ou outros processos educacionais. Para mais detalhes da Plataforma Arouca, acessar o sítio http://www.unasus.gov.br/plataforma_arouca (acessada em 21 de janeiro de 2013).

A Plataforma Arouca fornecerá aos usuários um módulo de análise de dados. As ferramentas desenvolvidas para este módulo deverão ser capazes de fornecer meios para a construção de mapas que agreguem os mais variados tipos de informações. Por exemplo, mapas que

permitem visualizar origem e destino dos profissionais de saúde de forma clara e objetiva e mapas de concentração de profissionais. O funcionamento destas ferramentas deverá ser *online*, uma vez que a Plataforma opera *online* na web.

Neste trabalho propomos uma ferramenta de análise de dados construída em linguagem R para ser incorporada no módulo de análise de dados da Plataforma Arouca. Esta ferramenta constitui de um pacote composto por funções que permitem ao usuário explorar dados georreferenciados a partir de interfaces gráficas interativas. Este pacote foi nomeado por `aRouca` . Embora os mapas sejam construídos via R , os mesmos são exportados para linguagem `KML` ou `SVG` . Desta forma, a visualização dos mapas é feita através do aplicativo Google Earth/Maps para a linguagem `KML` e pelo navegador (*browser*) web para arquivos com a linguagem `SVG` .

O pacote `aRouca` foi projetado para operar junto à Plataforma Arouca. Entretanto, pode ser utilizado independente da Plataforma. Ou seja, as funcionalidades do `aRouca` estarão disponíveis tanto para os usuários da Plataforma Arouca, quanto para os usuários comum do R .

3.2 Funcionalidades

O pacote `aRouca` oferece funções para análise exploratória de dados georreferenciados. As funções implementadas neste pacote fornece meios para o desenvolvimento de mapas temáticos, tais como mapas de áreas, pontos proporcionais e mapas para visualizar fluxo do tipo origem/destino. Além disso, o pacote oferece ferramentas para o cálculo de taxas brutas e suavizadas (bayesianas), e utilizar estas taxas como opção de tema para os mapas. Estas funções e outras serão apresentadas na Seção 3.2.1.

Na construção de mapas temáticos é importante classificar a informação que será visualizada no mapa, desta forma o pacote fornece dois tipos de classificação: um através dos quantis da informação desejada e outra dividindo em classes iguais. Além disso, o pacote fornece algumas alternativas para seleção das cores a serem utilizadas na customização dos mapas.

Outra funcionalidade do pacote `aRouca` é a opção de salvar o mapa temático como ar-

quivo PNG. Esta funcionalidade auxilia o usuário que esteja interessado em montar relatórios de suas análises. Além disso, os mapas salvos tem a mesma imagem que é exibida dentro do Google Maps, tornando o mapa mais elegante.

Como citado anteriormente na Seção 2.1.1, o pacote aRouca utilizará o formato de arquivo *shapefile* para representar estrutura espacial dos mapas. Desta forma, utilizará métodos e classes definidos pelo pacote sp para representar e manipular estruturas espaciais no R . Os mapas disponíveis no pacote são: municípios, micro regiões, macro regiões, unidades federativas e regionais de saúde do Brasil. O arquivo *shapefile* com a estrutura espacial dos municípios foi obtido através do sítio do Instituto Brasileiro de Geografia e Estatística (IBGE) e os demais mapas foram construídos a partir dele.

3.2.1 Aplicação 1 - Dados de Áreas

Nesta seção apresentaremos um estudo de caso com algumas aplicações das funcionalidades do pacote aRouca . Abaixo segue um breve tutorial ilustrando como utilizar as funcionalidades do pacote.

Antes de usar o pacote é necessário instalar o software R e alguns pacotes. O download do R pode ser feito através do seu próprio sítio, http://cran.br-r-project.org (acessada em 05 de agosto de 2012). A instalação dos pacotes via linha de comando é exemplificada abaixo. Entretanto, existe a alternativa de instalar os pacotes utilizando a interface do R . Como a maioria dos pacotes possui dependências de vários outros e portando, usa-se o argumento dep = TRUE que faz com que todos os demais pacotes necessários sejam também instalados.

```
1 install.packages(c("maptools", "maps", "classInt", "plotrix", "descr", "RColorBrewer", "spdep", "
    Cairo", "XML", "RgoogleMaps", "PBSmapping"), dep = TRUE)
2 install.packages(c("SVGAnnotation", "RKML"), repos = "http://www.omegahat.org/R", type = "source"
    , dep = TRUE)
```

O pacote aRouca não está disponível no CRAN, portando a instalação não pode ser feita através da internet. A instalação deve ser feita a partir do arquivo zip, para usuários do sistema operacional Windows ou tar.gz para usuários do Linux. Estes arquivos podem ser

baixados através dos sítios http://goo.gl/sfDR1 ou http://goo.gl/fmpuh, respectivamente. Para a instalação dos pacote `aRouca` o usuário terá que fornecer o nome completo do arquivo, como no exemplo:

```
1 install.packages("~/R/aRouca.zip", repos = NULL)
2 library(aRouca)
```

Após a instalação do pacote, o próximo passo é carregá-lo através do comando `library()` ou `require()`, como exemplificado na linha 2 do exemplo acima. Para acessar os mapas disponíveis no pacote, basta utilizarmos a função `data()` e nome do objeto com a estrutura espacial, como exemplificado abaixo.

```
1 data(shape.file.mun)       #shape dos municípios
2 data(shape.file.micro)       #shape das microrregiões
3 data(shape.file.uf)        #shape das UF's
4 data(shape.file.meso)      #shape dos mesorregiões
5 data(shape.file.region)      #shape das regiões
6 data(shape.file.regioes.saude)  #shape das regionais de saúde
```

Os dados utilizados neste exemplo são disponibilizados pelo Departamento de Informática do SUS - DATASUS e podem ser acessados pelo sítio datasus.gov.br (acessado em 24 de janeiro de 2013). O exemplo a seguir são de óbitos ocorridos por câncer de Brônquios e Pulmões nas unidades federativas do Brasil e estão estratificados por sexo. O acesso a esta base de dados deve ser como segue o exemplo:

```
1 data(UFex)
2 head(UFex)
```

Algumas restrições são feitas para que as funções do pacote `aRouca` possam ser utilizadas. Uma delas é a dimensão do banco de dados, que deve ter no mínimo 4 colunas e sem limite para o número de linha. Sendo as colunas do banco de dados disposta da seguinte forma:

Identificador	Rótulo	Atributo	População	Opcional
Código que identifica a região (municípios, estados, *etc.*) de interesse, de acordo a com a identificação utilizada nos *shapefiles*.	Nome da respectiva região. Ex.: Amazonas.	Variável que o usuário deseja visualizar no mapa	População referente para o cálculo de taxas	Informação adicional que o usuário deseja exibir dentro dos balões.

Tabela 3.1: *Estrutura do banco de dados para utilizar funções para geração de arquivos* `KML`.

Para melhor compreender a Tabela 3.1 basta utilizar o comando `head()` especificado na segunda linha do código que exibirá a estrutura da base da dados que vamos utilizar no estudo de caso.

Como a codificação de caracteres utilizada pelo Linux é UTF-8, enquanto que do Windows é WINDOWS-1252, algumas funções podem do R podem falhar. Além disso, se não utilizarmos a codificação correta, os caracteres ficarão ilegíveis. Portanto, utilizaremos a função `toUTF8` para recodificar os caracteres para UTF-8, como no exemplificado a seguir.

```
1 data(shape.file.uf) #carregando o arquivo shapefile das unidades federativas.
2 head(shape.file.uf@data) #consultado o slot com as informações dos estados.
3 shape.file.uf$NOME <- toUTF8(shape.file.uf$NOME, "UTF-8") #recodificar os caracteres para UTF-8.
4 print(shape.file.uf@data)
```

Para construção de um arquivo `KML` para ser visualizado no Google Earth/Maps foram criadas funções separadas para cada nível regional. Por exemplo, para criação de mapas de áreas temáticos temos as seguintes funções: `gera_kml_poly_meso`, `gera_kml_poly_micro`, `gera_kml_poly_municipio`, `gera_kml_poly_regiao`, `gera_kml_poly_rsaude` e `gera_kml_poly_uf`. Estas funções permitem a construção de mapas temáticos para as mesorregiões, microrregiões, municípios, regiões, regionais de saúde e para unidades federativas do Brasil, respectivamente. Na Tabela 3.2 são especificados os argumentos destas funções.

Como no estudo de caso estamos lidando com um banco de dados composto pelos estados brasileiros, a função que devemos utilizar é a `gera_kml_poly_uf` para criarmos um mapa temático de polígonos, sendo os polígonos as unidades federativas. Abaixo segue o exemplo:

```
1 gera_kml_poly_uf(file.name = UFex, num.faixas = 5)
```

Argumentos	Descrição	Opções	Padrão
file.name	Nome do objeto em R que contém o banco de dados com as informações a serem visualizadas no mapa.	Argumento obrigatório.	NULL
cor	Cor que será utilizada para pintar o mapa temático. Argumento númerico inteiro.	(1) Blues, (2) Greys, (3) Greens, (4) OrRd, (5) YlOrRd, (6) PubuGn, (7) Oranges	2
m	Valor que irá multiplicar a taxa calculada.	Qualquer valor numérico.	10000
cut	Tipo de classificação dos intervalos para a informação do mapa.	(0) Amplitude, (1) Quantil.	1
tx	Tipo de taxa a ser calculada.	(0) Taxa bruta, (1) Taxa bayesiana global, (2) Taxa bayesiana local	0
num.faixas	Número de intervalos para classificar a informação.	Valores inteiros até 7.	4
borda.mapa	Permite visualizar os limites dos polígonos	(0) Com borda, (1) Sem borda.	1
r	Número de casas decimais para arredondamento dos valores.	Qualquer valor numérico inteiro.	2
graf	Tipo de gráfico a ser exibido no balão (description).	(“graf”) sem gráfico, (“bar”) gráfico de barras, (“pie”) gráfico de setores	“graf”
legenda.vis	Parâmetro de visualização da legenda.	(TRUE) legenda é visualizada, (FALSE) legenda não é visualizada.	TRUE
fixed	Parâmetro de classificação do atributo de acordo que o usuário desejar.	Qualquer vetor numérico com dimensão maior que 1.	NULL

Tabela 3.2: *Argumentos disponíveis nas funções de geração de mapas temáticos de áreas. Entre parênteses está o valor a ser utilizado como parâmetro do argumento.*

O único argumento obrigatório para utilizar a função `gera_kml_poly_uf` é o *file.name*. Desta forma, basta especificarmos apenas este argumento que a função já estará pronta para ser utilizada. Os outros parâmetros opcionais serão configurados com os valores padrões definidos na quarta coluna da Tabela 3.2. As funções citadas para construção de mapas de áreas tem como *output* dois arquivos, sendo um contendo o código `KML` para ser visualizado no Google Earth/Maps com extensão `KML` e o outro é uma imagem `png` com a legenda do mapa. Para que a legenda seja exibida no Google Earth/Maps é necessário que a imagem `png` esteja no mesmo diretório que o arquivo `KML` . A Figura 3.1 apresenta o resultado da função `gera_kml_poly_uf` com todos os argumentos opcionais configurados como padrões.

Pela Tabela 3.2 observamos que temos sete opções de cores e que o valor configurado como padrão é uma escala de cinza (`cor = 2`), que também podemos ver pela Figura 3.1.

Figura 3.1: *Exemplo da função* `gera_kml_poly_uf()` *para criação de mapa de áreas das unidades federativas do Brasil. Todos os argumentos opcionais estão configurados com os valores padrões, exceto o argumento* `num.faixas = 5`.

Além disso, dentro do balão anexado ao polígono correspondente ao estado do Pará, temos a informação exata da taxa de óbitos por câncer de pulmão e o tamanho da população referente a este estado. Porém, a forma que apresentamos não é muito elegante, pois há valores de taxa menores que um. Uma forma para contornar este problema é aumentar o valor do parâmetro `m`, que é responsável por multiplicar o valor da taxa calculada.

Outra opção para inserir mais informação no mapa é utilizar a tag `<description>` oferecida pela linguagem `KML`. Esta tag permite a inserção de códigos HTML, link de alguma página, ou até mesmo vídeos. Na composição do pacote `aRouca` utilizamos a tecnologia Google Charts API, que permite a construção de gráficos através de uma chamada URL, ver Seção 2.3. Desta forma, podemos integrar a linguagem `KML` ao Google Chart API através da tag `<description>`. Esta integração é feita inserindo a URL do Google Chart API, correspondente a construção do gráfico, dentro da tag `<description>`. Abaixo apresentamos como configurar a função para inserir gráfico de setores ou de barras dentro do balão e aumentar o valor argumento `m`.

```
1 gera_kml_poly_uf(file.name = UFex, num.faixas = 5, graf = "pie", m = 100000)
2 gera_kml_poly_uf(file.name = UFex, num.faixas = 5, graf = "bar", m = 100000)
```

Os comandos acima geram as Figuras 3.2 e 3.3, respectivamente. No pacote `aRouca` fornecemos estas duas opções de gráficos, porém outros tipos de gráficos podem ser implementados facilmente, como explicado na Seção 2.3. A vantagem desta integração é que não precisamos salvar os gráficos para serem apresentados no balão, pois estes são construídos apenas quando o usuário clica no polígono, poupando assim espaço no disco rígido. Entretanto, para utilizar esta integração é necessário estar conectado a internet, uma vez API do Google precisa consultar um página web.

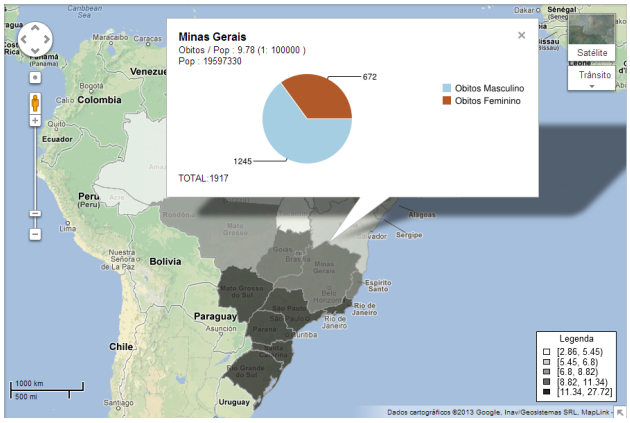


Figura 3.2: *Gráfico de setores.*

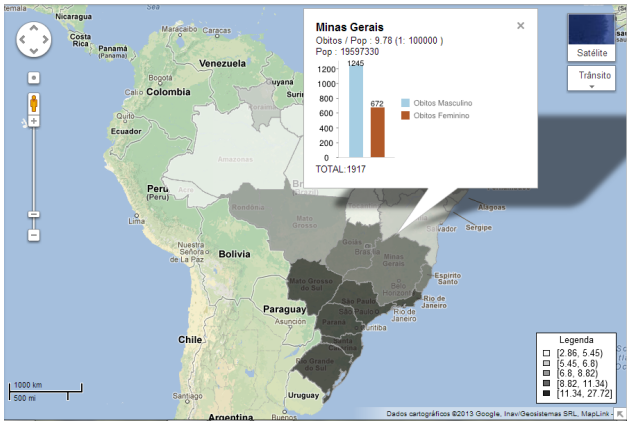


Figura 3.3: *Gráfico de barras.*

Figura 3.4: *Exemplo da função* `gera_kml_poly_uf()` *para criação de mapa com o gráfico de setor(Figura 3.2) ou barra (Figura 3.3) dentro do balão e* `m = 100000`.

Os gráficos visualizados dentro do balão foram construídos a partir das variáveis *Óbitos Masculino* e *Óbitos Feminino* que fazem parte do base de dados *UFex*. Portanto, para as construção dos gráficos é necessário que os atributos que irão compor o gráfico estejam na quinta coluna da base de dados em diante, sendo assim, construídos um gráfico para cada estado.

No caso em que o usuário deseja visualizar no mapa apenas uma contagem e não uma taxa devemos fazer uma modificação no banco de dados. Esta modificação é realizada substituindo os valores referentes as populações de cada estado por `NA`, como no exemplo que segue:

```
1 pop <- UFex$Pop #guardando o tamanho das populações no objeto pop
2 UFex$Pop <- NA
3 gera_kml_poly_uf(file.name = UFex, num.faixas = 5, graf = "pie", cor = 1)
4 gera_kml_poly_uf(file.name = UFex, num.faixas = 5, graf = "bar", cor = 3)
```

Os comandos utilizados acima tem como resultados as Figuras 3.5 e 3.6 e exemplificam as opções de cores do pacote `aRouca` , bem como a construção de mapas para variáveis que são contagens.

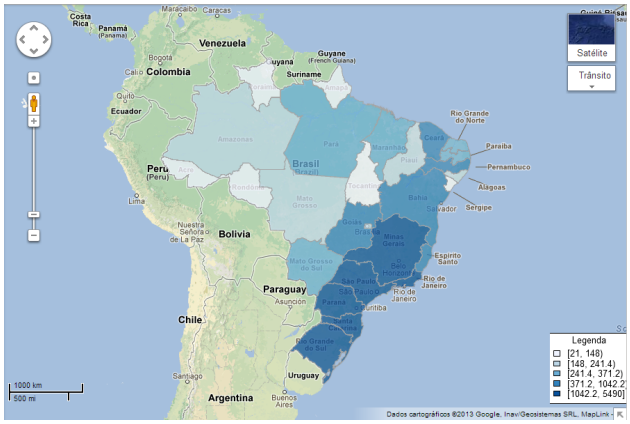


Figura 3.5: *Argumento de* `cor=1`.

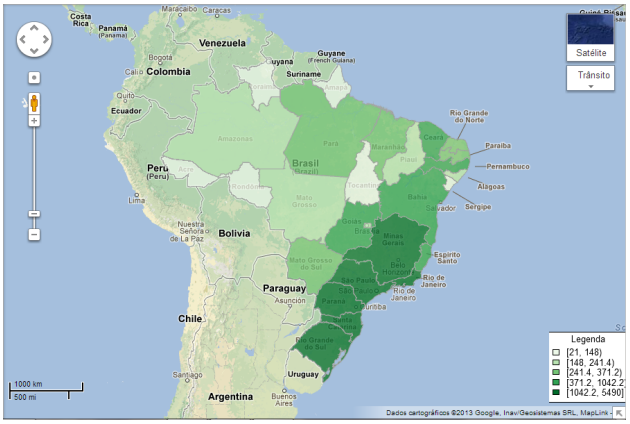


Figura 3.6: *Argumento de* `cor=3`.

Figura 3.7: *Exemplo da função* `gera_kml_poly_uf()` *configurada para gerar mapas temático com as cores azul e verde, respectivamente.*

Para exemplificarmos o cálculo de taxas bayesianas, iremos utilizar dados dos municípios de Minas Gerais, Brasil. Os dados referentes a estes municípios foram retirados também do DATASUS e são ocorrências de óbitos infantis no ano de 2010. Neste exemplo mostramos como realizar o cálculo de mortalidade infantil a partir de funções do pacote `aRouca` . Esta taxa é uma estimativa do risco de um nascido vivo morrer durante o seu primeiro ano de vida. Além disso, este é um tipo de taxa que pode refletir as condições de saúde, socieconômica e de vida de uma população. Segundo informações do DATASUS, as taxas de mortalidade são geralmente classificadas em *altas* (50 ou mais), *médias* (20 - 49) e *baixas* (menos de 20).

O método para calcular a taxa de mortalidade infantil de maneira direta (taxa bruta), é a seguinte:

$$\frac{\textit{número de óbitos de residentes com menos de um ano de idade}}{\textit{número total de nascidos vivos de mães residentes}} \times 1.000.$$

Porém este tipo de cálculo apresenta problemas quando áreas geográficas possuem populações pequenas sob risco e quando o número de casos observados é muito baixo. O principal problema ocasionado são estimativas com pouca representatividade e com alta variabilidade. Marshall (1991), propôs uma solução para este problema por meio do método em-

pírico Bayes, e utilizando a localização geográfica do município como informação adicional, resultando em estimativas com menos variabilidade.

```
1 data(MUNex)
2 data(shape.file.municipio)
3 shape.file.municipio$NOME <- toUTF8(shape.file.municipio$NOME, "UTF-8")
4 gera_kml_poly_municipio(file.name = MUNex, tx = 0, graf = "bar", cor = 1, m = 1000, fixed = c
      (0,20,50,100)) #Cáculo de taxa bruta com os intervalos fixados.
5 gera_kml_poly_municipio(file.name = MUNex, tx = 2, graf = "bar", cor = 1, m = 1000, fixed = c
      (0,20,50,100)) #Cáculo de taxas bayesianas com os intervalos fixados.
```

Pelo resultado dos comandos listados acima podemos notar visualmente a diferença entre as duas taxas pela Figura 3.10. Nota-se que a taxa bruta apresenta uma variabilidade alta, sugerindo que a mortalidade infantil se distribui geograficamente de forma aleatória. Entretanto, pela Figura 3.9 que apresenta as taxas bayesiana, observamos alguns padrões geográficos. Por exemplo, no norte e nordeste de Minas Gerais e próximo a Zona da Mata.

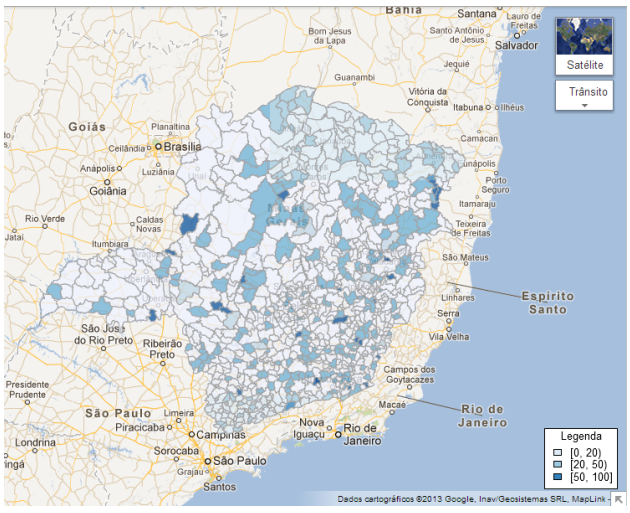


Figura 3.8: *Argumento* `tx=0`.

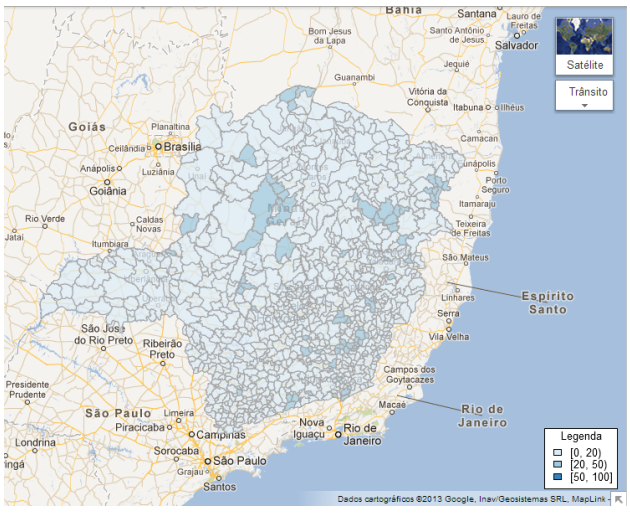


Figura 3.9: *Argumento* `tx=2`.

Figura 3.10: *Exemplo da função* `gera_kml_poly_municipio()` *configurada para calcular taxas brutas e bayesianas para mortalidade infantil, respectivamente.*

Nesta seção apresentamos como construir mapas temáticos para dados de áreas utilizando o pacote `aRouca` . Além disso, como utilizar as funcionalidades do Google Earth/Maps para inserirmos mais informações ao mapas através dos balões (`description`).

3.2.2 Aplicação 2 - Dados pontuais

O pacote `aRouca` permite também a construção de mapas de pontos proporcionais que são amplamente utilizados para visualizar geograficamente dados de eventos pontuais. A função `gera_kml_bubble_latlong` foi implementada para criar este tipo de mapa temático, e para utilizá-la é necessário que a base de dados tenha o seguinte formato:

Latitude	**Longitude**	**Rótulo**	**Atributo**	**População**	**Opcionais**
Latitude do ponto que deseja visualizar.	Longitude do ponto que deseja visualizar.	Rótulo do ponto. Por exemplo, UFMG.	Variável que o usuário deseja visualizar no mapa	População referente para o cálculo de taxas	Informação adicional que o usuário deseja exibir dentro dos balões.

Tabela 3.3: *Estrutura do banco de dados para utilizar a função* `gera_kml_bubble_latlong` *para geração de mapas com pontos proporcionais.*

A coluna *População* só é preenchida quando o usuário desejar algum tipo de taxa, caso contrário esta coluna deve ser preenchida por `NA`. Ou seja, o banco de dados dever ter no mínimo cinco colunas. As colunas que estiverem na sexta posição em diante serão utilizadas para criar os gráficos de barras ou de setores. Os argumentos da função `gera_kml_bubble_latlong` são os mesmos apresentados na Tabela 3.2. Abaixo segue o código de como produzir um mapa de pontos proporcionais.

```
1 data(pontoex)
2 head(pontoex)
3 gera_kml_bubble_latlong(file.name = pontoex, graf = "bar", cor = 1)
```

Os dados utilizados neste exemplo foram obtidos através da Plataforma Arouca e são dados de profissionais de medicina graduados em Belo Horizonte entre os anos 2000 e 2005. A variável de interesse neste exemplo são os destinos mais procurados pelos profissionais de medicina que formam nos anos citados. O resultado dos comandos acima é apresentado na Figura 3.11.

Além de visualizar a dispersão geográfica dos destinos e a quantidade de médicos que migraram para determinada cidade, o usuário ainda pode clicar em um dos pontos e visualizar

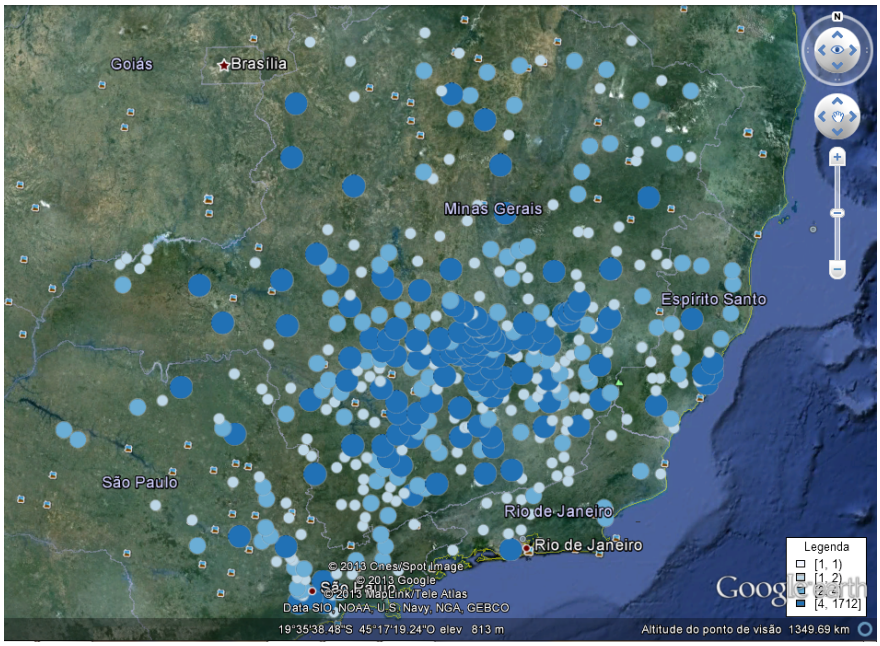


Figura 3.11: *Exemplo da função* `gera_kml_bubble_latlong()`

o gráfico de barras dentro do balão, como apresentado na Figura 3.12. Desta forma, o usuário tem como informação a quantidade de médicos que migraram em cada um dos anos, como exemplificado no balão.

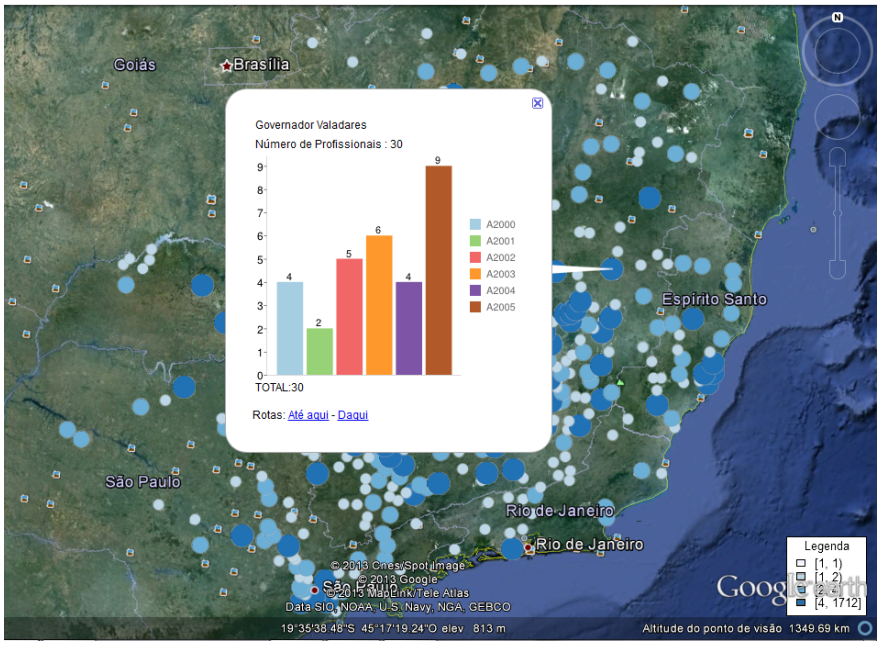
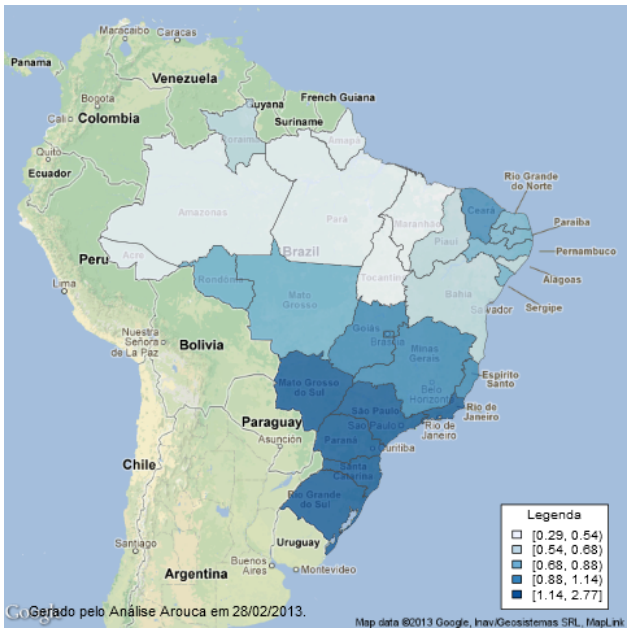


Figura 3.12: *Exemplo da função* `gera_kml_bubble_latlong()`*, configurada para visualizar o gráfico de barras.*

Outra funcionalidade implementada no pacote `aRouca` , é a opção de salvar os mapas em arquivos de imagem com a extensão `.png` e com o mesmo plano de fundo do Google Maps. Esta funcionalidade foi motivada pelos os usuários da Plataforma Arouca, que tinham

como objetivo salvar suas análises e incorporá-las em seus relatórios. Os comandos abaixo exemplificam o uso desta função.




```
1  data(shape.file.uf)
2  data(UFex)
3  SalvaPNG_poly_uf(file.name = UFex,
       cor = 1, num.faixas = 5)
```

Figura 3.13: *Estrutura hierárquica da linguagem* `XML` .

Esta funcionalidade está disponível apenas para os mapas de polígonos. As funções para os demais mapas são as seguintes: `SalvaPNG_poly_meso`, `SalvaPNG_poly_micro`, `SalvaPNG_poly_mun`, `SalvaPNG_poly_rsaude`, `SalvaPNG_poly_uf`. Os argumentos destas funções são os mesmos da Tabela 3.2, exceto pelo argumento de geração de gráficos, `graf`.

3.2.3 Aplicação 3 - Mapa de Origem/Destino

Uma outra funcionalidade implementada no pacote é a criação de mapa para visualizar fluxo do tipo origem-destino. Diferente das outras funcionalidade já citadas neste trabalho, a função para gerar mapa de origem-destino é baseada na linguagem `SVG` . A escolha de utilizarmos esta linguagem é pelo fato desta permitir a criação de imagens vetoriais e a inserção de códigos JavaScript. Esta característica da linguagem `SVG` de integrar códigos JavaScript nos permite fornecer aos usuários mapas interativos.

O formato da base de dados para criar o mapa de origem/destino é exemplificado abaixo:

A primeira coluna do banco de dados exemplificado pela Tabela 3.4 é composto pelo

Origem	Destino	RS	SC	PR
MG	NA	10	8	33
SP	NA	23	43	9
RJ	NA	56	21	11
SP	NA	22	12	30

Tabela 3.4: *Estrutura do banco de dados para criação de mapa de origem/destino.*

código da região de origem, sendo a primeira linha desta coluna composta pelo título desejado para o mapa de origem, que no exemplo acima é "Origem". A segunda coluna é utilizada para que o usuário posso inserir o título do mapa de destino, sendo todas as outras linhas desta coluna composta por `NA`. Da terceira coluna em diante, entramos com os dados, porém a primeira linha é composta pelos códigos dos destinos e as demais pelos dados. Por exemplo, na Tabela 3.4 exemplificamos que dez pessoas saíram do estado de Minas Gerais e migraram para o Rio Grande do Sul.

As seguintes funções foram implementadas no pacote `aRouca` para criação de mapas de origem/destino: `gera_origem_destino_municipio`, `gera_origem_destino_uf`, `gera_origem_destino_micro`, `gera_origem_municipio_meso` e `gera_origem_destino_rsaude`. O resultado destas funções é um arquivo com a extensão `.svg` e este formato de arquivo pode ser visualizada em qualquer navegador. Para mais detalhes sobre o formato `SVG`, ver a Seção 2.2.1 e a referências citas nela.

As funções listadas acima tem os mesmos argumentos da Tabela 3.2, exceto pelo argumento `graf` para criação de gráficos. Abaixo exemplificamos como utilizar umas das funções acima para gerar o mapa de origem/destino.

```
1 data(od)
2 head(od)
3 data(shape.file.uf)
4 gera_origem_destino_uf(file.name = od, cor = 1)
```

A interatividade deste mapa é feita ao clicar no estado de origem, que neste exemplo é o estado de Minas Gerais, e como resultado o mapa de destino é colorido segundo o número de indivíduos que saíram do estado de origem e que foram para cada um dos estados de destinos. Pela Figura 3.14 podemos visualizar o resultado dos comandos acima e observar

que o estado clicado pelo usuário muda para cor cinza, enquanto que os demais permanecem com a cor vermelha. Observamos também, que ao ficar com o cursor do mouse em cima do estado abrirá um balão (tooltip) com nome da região e com o respectivo valor. Além disso, foi implementado uma função para darmos *zoom* no mapa, esta função é acionada quando clicamos no objeto que está no canto superior esquerda da Figura 3.14. Estes tipos de interatividades foram criados a partir de códigos JavaScript e estão disponíveis na pasta do pacote `aRouca` .

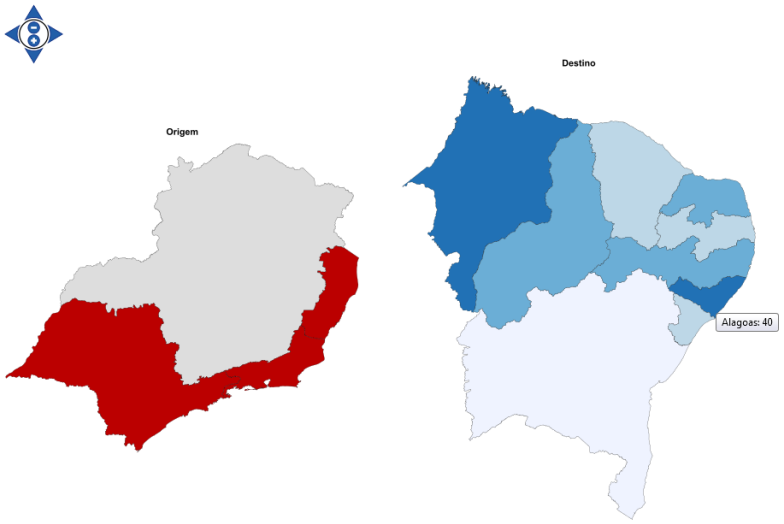


Figura 3.14: *Exemplo da função* `gera_origem_destino_municipio_uf`*, para gera mapa de origem/destino das UF's.*

Nesta seção foi apresentada as funcionalidades disponíveis para o usuário comum do R , através de exemplos que podem ser reproduzidos facilmente. Apresentamos também os mapas disponíveis no pacote, bem como os bancos de dados utilizados nos exemplos. Vale ressaltar que as funções não exemplificadas podem ser utilizadas de forma similar aos exemplos apresentados.

3.3 Integração do pacote `aRouca` à Plataforma Arouca

A integração do pacote `aRouca` à Plataforma Arouca é realizada utilizando o pacote `Rserve`, desenvolvido por Urbanek (2012). Este pacote atua como um servidor, permitindo que outros programas enviem solicitações ao R . As implementações no pacote `Rserve` estão disponíveis para várias linguagens populares, tais como C/C++ e Java.

A Plataforma Arouca utiliza, predominantemente, a linguagem Java em sua composição. Abaixo segue o trecho de código em Java que ilustra a comunicação da Plataforma Arouca com R e, consequentemente como o pacote aRouca . Para a execução remota do R é necessário executá-lo inicialmente em um terminal de comando e fazer a chamada ao Rserve(). O comando R CMD Rserve pode ser executado diretamente via terminal e implementa a ativação conjunta do R e Rserve(). A partir deste procedimento é possível executar funcionalidades do R a partir de outras linguagens, como Java.

```
1 RConnection connection = null;
2     try{
3       //Cria uma nova conexão com R
4       conexao = new RConnection();
5       REXP ret = conexao.parseAndEval("chamada de função");
6       files = ret.asStrings();
7     }
```

Pelo trecho de código acima, observamos que para enviarmos uma solicitação para R , devemos criar uma conexão com o servidor. Cada conexão requisitada possui a sua própria área de trabalho no R , evitando assim conflitos na presença de múltiplos usuários. Utilizando a função parseAndEval podemos executar qualquer função que esteja disponível no R . As funções implementadas no aRouca retornam o endereço em memória dos arquivos gerados. Quando os arquivos gerados pelo pacote aRouca são do tipo KML , estes são visualizados no ambiente Arouca a partir de API do Google Maps. Quando são arquivos do tipo SVG , os mesmos são incorporados na Plataforma Arouca em formato HTML.

A Plataforma Arouca foi projetada para acesso simultâneo de múltiplos usuários. Como consequência, as funções do pacote aRouca foram projetadas para gerar arquivos de saída cujos nomes são definidos aleatoriamente. Dessa forma, conflitos com relação aos acessos aos arquivos gerados são evitados.

Com relação à entrada de dados para as funções do R , os arquivos de entrada são gerados pelo usuário a partir de requisições específicas, diretamente na interface gráfica da Plataforma Arouca. Nesta interface, o usuário também especifica os parâmetros para a visualização dos mapas. É parte operacional da Plataforma Arouca, a geração dos banco de

dados que serão utilizado pelas funções do pacote `aRouca` e a especificação dos parâmetros a serem aplicados.

É importante destacar que, para um usuário da Plataforma Arouca, as operações de execução do ambiente R são ocultas, ou seja, a percepção pelo usuário é a operacionalização de todas as funcionalidades em um único ambiente computacional.

Capítulo 4

Conclusões

Esta dissertação de mestrado teve como objetivo o desenvolvimento de um pacote em ambiente R denominado `aRouca` , propondo novas visualizações georreferenciadas aplicadas à avaliação de dados de residência, formação e atuação de profissionais de saúde. A principal contribuição deste trabalho foi a integração de linguagens e ambientes de visualização como as linguagens `SVG` , `KML` , API's do Google e a linguagem R . Essas linguagens e ambientes foram agregados em um conjunto de funcionalidades para visualização de dados georreferenciados. Dessa forma, foram desenvolvidos novos recursos como a visualização de gráficos de barras, gráficos de setores e imagens integradas na linguagem `KML` . Em particular, foram utilizados recursos do Google Charts API para a geração dinâmica de imagens como chamadas de url em arquivos `KML` .

Também foram propostos esquemas inéditos para a visualização de dados do tipo origem-destino. Estes dados são obtidos, por exemplo, a partir da análise temporal do local de formação e/ou residência de profissionais de saúde e sua atual área geográfica de atuação. É de interesse, por exemplo, verificar a capacidade de captação de profissionais de saúde em regiões geográficas específicas no território brasileiro. A principal implementação em linguagem `SVG` é a função `gera_origem_destino()` que permite a visualização de dois mapas dinâmicos: o primeiro representa as regiões de origem e o segundo as regiões de destino. A partir de seleções em sub-regiões no mapa de origem, a intensidade e o esquema de cores no mapa de destino são alterados de forma a visualizar informação relativa ao deslocamento dos profissionais de saúde. Esta aplicação em particular foi realizada em linguagem `SVG` em

virtude da praticidade de tal linguagem na manipulação interativa de dados e imagens.

É importante destacar que as funcionalidades desenvolvidas já estão em funcionamento junto à Plataforma Arouca. As principais bases de dados, funções e exemplos também foram encapsuladas no pacote `aRouca` , o que permite que novos usuários possam acessar as suas funcionalidades fora da Plataforma Arouca.

Recentemente, novos pacotes foram apresentados na literatura, como os pacotes `plotKML` e `RKML`. Como atividades futuras, pretende-se avaliar e propor novas visualizações utilizando esses recursos.

Referências Bibliográficas

Bivand *et al.* **(2012)** Roger Bivand, with contributions by Micah Altman, Luc Anselin, Renato Assunção, Olaf Berke, Andrew Bernat, Guillaume Blanchet, Eric Blankmeyer, Marilia Carvalho, Bjarke Christensen, Yongwan Chun, Carsten Dormann, Stéphane Dray, Rein Halbersma, Elias Krainski, Pierre Legendre, Nicholas Lewin-Koh, Hongfei Li, Jielai Ma, Giovanni Millo, Werner Mueller, Hisaji Ono, Pedro Peres-Neto, Gianfranco Piras, Markus Reder, Michael Tiefelsdorf e Danlin Yu. *spdep: Spatial dependence: weighting schemes, statistics and models*, 2012. URL http://CRAN.R-project.org/package=spdep. R package version 0.5-46. Citado na pág.

Bivand *et al.* **(2008)** Roger S. Bivand, Edzer J. Pebesma e Virgilio Gomez-Rubio. *Applied spatial data analysis with R*. Springer, NY. URL http://www.asdar-book.org/. Citado na pág. 13

Blum (1948) H. F. Blum. Sunlight as a causal factor in cancer of the skin of man. *Journal of the National Cancer Institute*, 9:247–258. Citado na pág. 5

Friedman (2008) V. Friedman. Data visualization and infographics. *Graphics, Monday Inspiration*. Citado na pág. 2

Friendly (2009) M. Friendly. Milestones in the history of thematic cartography, statistical graphics, and data visualization. Relatório técnico, York University, Canada. Citado na pág. 2, 3

Google (2012a) Google. *Google Earth*. http://www.google.com.br/intl/pt-BR/earth/index.html, 2012a. Citado na pág. 6

Google (2012b) Google. *Google Maps*. http://maps.google.com.br/, 2012b. Citado na pág. 6

Google (2012c) Google. *Referência do KML*, 2012c. URL https://developers.google.com/kml/documentation/kmlreference?hl=pt-BR. Citado na pág. 20

Hornik (2012) Kurt Hornik. The R FAQ, 2012. URL http://CRAN.R-project.org/doc/FAQ/R-FAQ.html. ISBN 3-900051-08-9. Citado na pág. 11

IBM (2012) IBM. *Many Eyes*. http://www-958.ibm.com/software/data/cognos/manyeyes/, 2012. Citado na pág. 7

Lang (2012a) D. Lang, D. T. e Nolan. *RKML: Simple tools for creating KML displays from R*, 2012a. R package version 0.8-0. Citado na pág. 6

Lang (2012b) D. Lang, D. T. e Nolan. *SVGAnnotation: Tools for post-processing SVG plots created in R*, 2012b. R package version 0.91-0. Citado na pág. 6, 15

Lang (2012c) D. T. Lang. *XML: Tools for parsing and generating XML within R and S-Plus.*, 2012c. URL http://www.omegahat.org/RSXML. R package version 3.93-0. Citado na pág. 6, 14

Lewin-Koh (2012) R. Lewin-Koh, N. J. e Bivand. *Package maptools: Tools for reading and handling spatial objects*, 2012. URL http://CRAN.R-project.org/package=maptools. R package version 0.8-14. Citado na pág. 6, 13

Loecher (2012) M. Loecher. *Package RgoogleMaps: Overlays on Google map tiles in R.* http://cran.r-project.org/web/packages/RgoogleMaps/, 2012. URL http://CRAN.R-project.org/package=RgoogleMaps. R package version 1.2.0. Citado na pág. 6

Marshall (1991) R.J. Marshall. Mapping disease and mortality rates using empirical bayes estimators. *Journal of the Royal Statistical Society*, 40:283–294. Citado na pág. 7, 37

Nolan (2012) D. T. Nolan, D. e Lang. Interactive and animated scalable vector graphics and r data displays. *Journal of Statistical Software*, 1:3,43. Citado na pág. 6

Palm (1890) T. A. Palm. The geographical distribution and aetiology of rickets. *Practitioner*, 45:270–279. Citado na pág. 5

Pebesma e Bivand (2005) Edzer J. Pebesma e Roger S. Bivand. Classes and methods for spatial data in R. *R News*, 5(2):9–13. URL http://CRAN.R-project.org/doc/Rnews/. Citado na pág. 12

R Development Core Team (2012) R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. URL http://www.R-project.org/. ISBN 3-900051-07-0. Citado na pág. 13

Ribeiro (2011) Paulo Justiniano Ribeiro. *Introdução ao Ambiente Estatístico R*, maio 2011. URL http://www.leg.ufpr.br/~paulojus/embrapa/Rembrapa/. Citado na pág. 12

Rocha (2012) H. Rocha. *SVG: Uma Introdução Prática*, 2012. Citado na pág. 19

Team (2012) R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. URL http://www.R-project.org/. ISBN 3-900051-07-0. Citado na pág. 6, 10

Tukey (1977) J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley. Citado na pág. 4

Ullman e Widom (1997) Jeffrey D. Ullman e Jennifer Widom. *A first course in database systems*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA. ISBN 0-13-861337-0. Citado na pág. 1

Unwin *et al.* (2006) A. Unwin, M. Theus e H. Hofmann. *Graphics of Large Datasets: Visualizing a Million*. Springer. Citado na pág. 1

Urbanek (2012) Simon Urbanek. *Rserve: Binary R server*, 2012. URL http://CRAN.R-project.org/package=Rserve. R package version 0.6-8. Citado na pág. 43

Young *et al.* (2006) F. W. Young, P. M. Valero-Mora e M. Friendly. *Visual statistics : seeing data with dynamic interactive graphics*. John Wiley & Sons, Inc., Hoboken, New Jersey. Citado na pág. 1, 2, 4