UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL
Tecnologia da Informação na Saúde

Manipulação de Base de Dados usando R

Objetivo

O objetivo desta lista de exercícios é aferir o seu conhecimento teórico e prático sobre as técnicas de manipulação de base de dados através do uso das linguagens de programação R, considerando diferentes fontes de dados e modos manipulação e visualização dos mesmos.

Introdução

Como discutido em aula, estudos que envolvam análises quantitativas, tipicamente, utilizam dados de diversas fontes. Por exemplo, um estudo epidemiológico pode envolver muito mais do que apenas os dados clínicos, mas também deve levar em conta dados que caracterizem a situação dos indivíduos que fazem parte do estudo, incluindo então, por exemplo, dados socioeconômicos. As bases de dados são disponibilizadas em diferentes formatos. Assim, você deve se preparar para obter os dados, trata-los e visualiza-los de diferentes formas. Aqui iremos explorar as técnicas de *webscraping* e visualização usando mapas.

Webscraping

Web Scraping (raspagem) é uma técnica de engenharia de software atualmente muito utilizada por empresas para extrair dados/conteúdos existentes em websites. De uma forma mais simples, é a tarefa de extrair dados de sites da internet de forma automatizada, sendo direta (usando o protocolo HTTP ou outro protocolo que se comunique diretamente ao servidor alvo) ou por meio de um *webbrowser*.

É uma ferramenta muito útil, pois facilita a busca de informações pela web e facilita a análise dos dados. E a maior vantagem são para aqueles que sabem codificar, pois linguagens com Python e R permitem manipular e visualizar estes dados de diferentes formas.

Como tudo tem um limite, existem regras que delimitam o uso e acesso dessa técnica. Cabe ressaltar que, se você fazer o uso de *scraping* em sites que não permitem o mesmo, isso pode ser considerado *hacking*, podendo ter implicações legais. Por isso o ideal é se certificar que aquilo que você pretende fazer é permitido e legal. Um artigo muito interessante sobre o tema pode ser encontrado através do endereço: http://observatoriodaimprensa.com.br/etica-jornalistica/os-limites-da-garimpagem-de-dados-na-internet/.

Para construir um web *scraper* geralmente é necessário estudar como o site a ser acessado foi construído, se tem limites de requisições, utilização de cookies, sessões etc; como e com que

frequência o site é atualizado, tanto em relação à sua interface como em relação aos dados que queremos extrair; qual o caminho percorrido para acessar uma página específica. A linguagem R oferece um conjunto diverso de pacotes para este fim, tais como pacote ***httr*** para fazer requisições HTTP através do R e baixar arquivos; além dos pacotes ***xml2***, ***rvest*** e ***jsonlite*** para obter informações estruturadas de arquivos .xml, .html e json. Nesta atividade usaremos o pacote ***rvest***. Documentação sobre este pacote pode ser consultada em https://cran.r-project.org/web/packages/rvest/rvest.pdf.

Neste exemplo, nosso objetivo é obter os dados populacionais para todos os municípios do Rio Grande do Norte. Para isso, obteremos os dados diretamente do site do Instituto Brasileiro de Geografia e Estatística (IBGE), seguindo o link usado a seguir.

Municípios	Código	Gentílico	População 2010	Área da unidade territorial 2016 (km²)	Densidade demográfica 2010 (hab/km²)	Série revisada - Valor adicionado bruto Total, a preços correntes
Acari	2400109	acariense	11.035	608,47	18,1	95.423
Açu	2400208	açuense	53.227	1.303,44	40,8	835.667
Afonso Bezerra	2400307	afonso-bezerrense	10.844	576,18	18,8	64.074
Água Nova	2400406	água-novense	2.980	50,68	58,8	20.795
Alexandria	2400505	alexandrinense	13.507	381,21	35,4	94.209
Almino Afonso	2400604	almino-afonsense	4.871	128,04	38,0	29.562
Alto do Rodrigues	2400703	alto-rodriguense	12.305	191,33	64,3	522.398

Figura 1. Conteúdo da página web do IBGE usada como alvo.

Dica: acesse o link através do seu browser para visualizar as informações que serão obtidas. Um resumo é mostrado na Figura 1.

Devemos iniciar com o downloading e parsing do conteúdo alvo, usando a função ***read_html()*** que retorna um documento XML contendo todas as informações da página web.

```
require(rvest) # importa a biblioteca rvest

cidades<-
read_html("http://cidades.ibge.gov.br/download/mapa_e_municipios.php?lang=&uf=rn")  %>%
html_table(fill=TRUE)
```

O próximo passo é limpar os dados obtidos, mantendo apenas aquilo que nos interessa. Para o nosso exemplo, iremos manter apenas o nome do município, CEP e a população em 2010.

```
library(stringr) #importa a biblioteca stringr

base<-data.frame(cidades[[1]][c(1,2,4)]) # Selecionando os dados de interesse
names(base)<-c("cidade","cep","pop2010") # Nomeando as colunas
row.names(base)<-1:168                   # Enumerando as linhas
base$pop2010<-str_replace_all(base$pop2010,"\\.","") # Retira os pontos

base$pop2010[is.na(base$pop2010)] <- 0   # Substitui valores NAs por zeros
base <- head(base,-1)                    # Remove a ultima linha da tabela

base$pop2010<-as.numeric(base$pop2010) # Converte os caracteres em numéricos

head(base)
```

Como resultado da operações acima, o comando head(base) irá produzir a seguinte saída:

```
          cidade     cep pop2010
1          Acari 2400109   11035
2            Açu 2400208   53227
3 Afonso Bezerra 2400307   10844
4      Água Nova 2400406    2980
5     Alexandria 2400505   13507
6  Almino Afonso 2400604    4871
```

Visualização usando mapas

O R tem muitos pacotes para plotar mapas, tanto estáticos quanto interativos. Somente para citar alguns voltados a mapas estáticos: sp, maptools, ggplot2, ggmap, maps, choropleth, rworldmap, GISTools e o tmap. Para mapas interativos, existem o ggvis, o googleVis, o leaflet, o highcharter e o tmap. Neste exemplo iremos focar no uso do sp, por sua simplicidade de uso, entretanto, os princípios apresentados aqui servirão de base para a utilização de qualquer dos outros pacotes citados.

Uma das formas mais comuns de manipulação de mapas, inclusive no R, é através do uso de estruturas de polígonos. Há um repositório de base de dados espaciais na internet chamado "*Global Administrative Areas*", que pode ser acessado por meio do link: ***gadm.org***. Este repositório permite a exportação dos polígonos em vários formatos, inclusive em data frame para o R. Além disso, os polígonos podem ser exportados em vários níveis administrativos: país, províncias, municípios e cidades. Após realizar o download dos polígonos para o Brasil, precisamos organizar o dataframe que contêm o polígono. O polígono a ser importado consiste no mapa do Brasil e respectivos e municípios.

A nossa tarefa agora inclui selecionar apenas os municípios do estado do Rio Grande do Norte e depois incorporar os dados sobre a população, obtidos do IBGE.

Para isso, o primeiro passo é a importação dos polígonos.

```
library(sp)

setwd("E:/aular/mapas") # Define o diretório de trabalho

br <- readRDS("BRA_adm2.rds") # Importa os polígonos do arquivo no diretório de trabalho
```

Já aqui podemos visualizar o mapa do Brasil.

```
# Plotando o mapa do Brasil, organizado por municípios
plot(br)
```

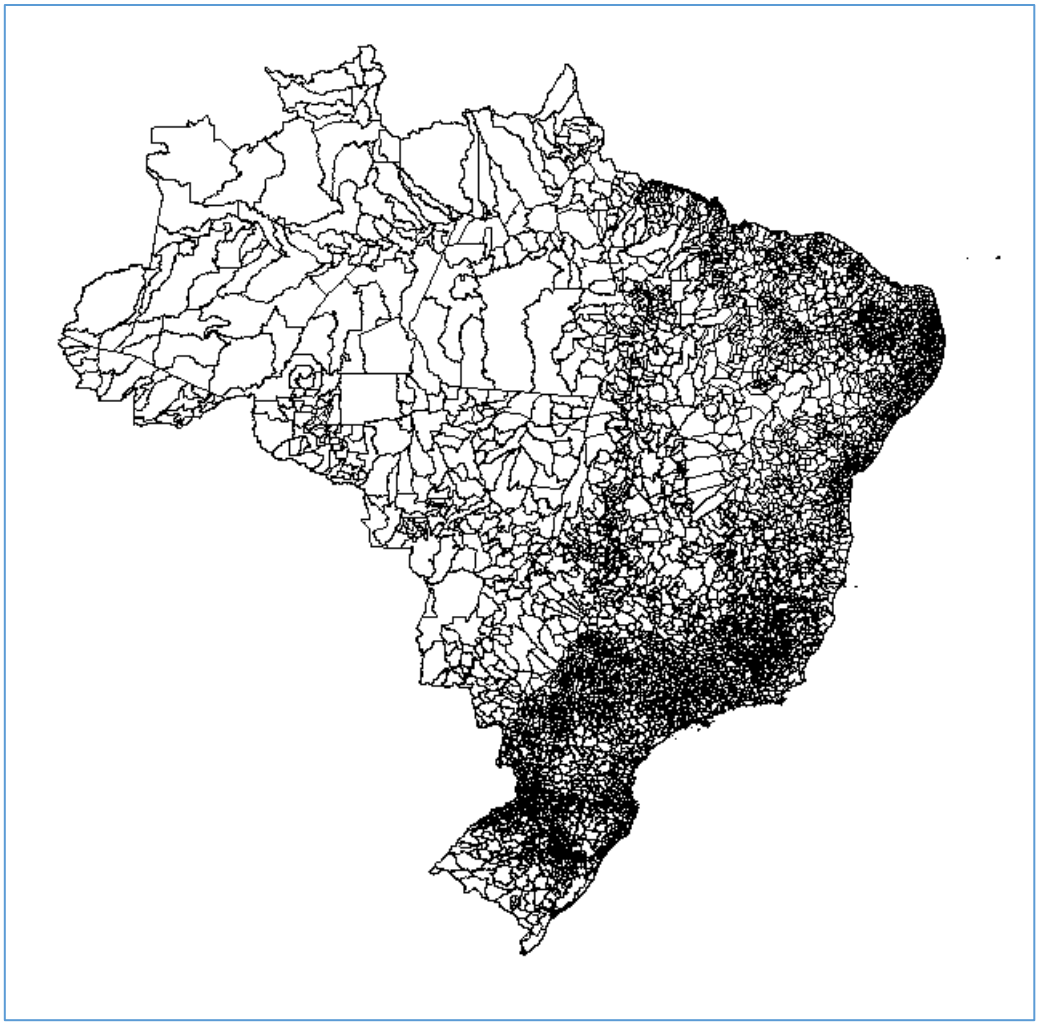

Figura 2. Mapa do Brasil organizado em municípios.

Após importar as definições dos polígonos para todos os municípios do Brasil, devemos filtrar os dados para obter apenas os municípios do estado do Rio Grande do Norte.

```
rn = (br[br$NAME_1=="Rio Grande do Norte",]) # Filtrando apenas os municípios do RN
```

Já aqui podemos visualizar o mapa do Rio Grande do Norte.

```
# Plotando o mapa do Rio Grande do Norte
plot(rn)
```

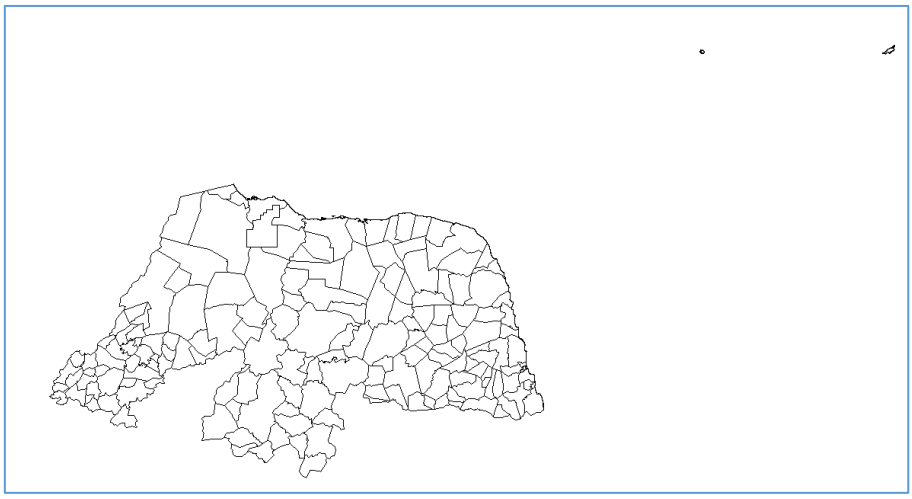

Figura 3. Mapa do Rio Grande do Norte organizado em municípios.

Usando a função ***plot()*** é possível plotar o mapa do RN e dar destaque ao município de Natal, como mostrado a seguir.

```
# Plotando o mapa do Rio Grande do Norte com a cidade de Natal destacada em vermelho
plot(rn)
plot(rn[rn$NAME_2=="Natal",], add=T, col="red")
```

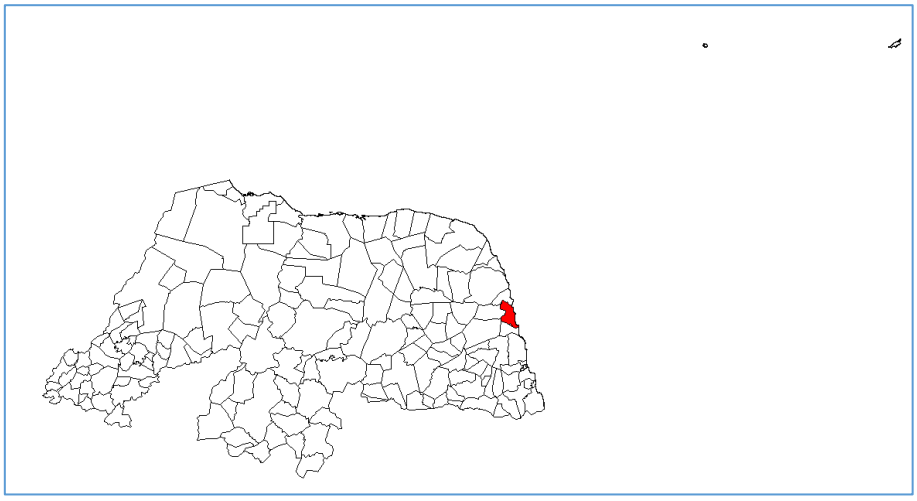

Figura 4. Mapa do Rio Grande do Norte organizado em municípios e com destaque para o município de Natal.

Agora que já temos o dataframe ***rn*** contendo os polígonos com a representação dos municípios do RN, podemos adicionar a eles as informações sobre a população que obtivemos do IBGE via *webscrapping*. Para isso, a solução óbvia em R é o uso da função ***merge()***. Entretanto, o seu uso normal pressupões que os dois dataframes de origem possuem um campo comum e com mesmo nome. Em nosso exemplo (e algo comum na vida real), os nomes dos campos diferem entre os dataframes.

Um cuidado a se tomar antes de um merge é inspecionar bem os seus dados a fim de identificar inconsistências que possam levar a um resultado incorreto do *merge*. Um dos problemas mais comuns é a escrita em um dataframe ser ligeiramente (ou completamente diferente) no outro dataframe e, assim, não haverá correspondência na operação de *merge*. Neste sentido, nossos dataframes apresentam a seguinte particularidade: os dados dos polígonos parecem bem desatualizados e apresentam dados de municípios que precisam ser corrigidos para que haja uma correspondência com os dados obtidos do IBGE (naturalmente mais recentes e atualizados). Abaixo, são apresentadas as correções necessárias, detectadas após exaustiva verificação dos dados (papel de um bom cientista de dados!).

```
rn$NAME_2[which(rn$NAME_2=="Governador Dix-Sept Rosad")]<-"Governador Dix-Sept Rosado"
rn$NAME_2[which(rn$NAME_2=="Lagoa de Anta")]<-"Lagoa d`Anta"
rn$NAME_2[which(rn$NAME_2=="Lagoas de Velhos")]<-"Lagoa de Velhos"
rn$NAME_2[which(rn$NAME_2=="Jardim-Piranhas")]<-"Jardim de Piranhas"
rn$NAME_2[which(rn$NAME_2=="Olho-d'Água do Borges")]<-"Olho-d`Água do Borges"
rn$NAME_2[which(rn$NAME_2=="Passabém")]<-"Passagem"
rn$NAME_2[which(rn$NAME_2=="Santana")]<-"Santana do Seridó"
rn$NAME_2[which(rn$NAME_2=="Junco")]<-"Messias Targino"
rn$NAME_2[which(rn$NAME_2=="São Miguel de Touros")]<-"São Miguel do Gostoso"
rn$NAME_2[which(rn$NAME_2=="Presidente Juscelino")]<-"Serra Caiada"
rn$NAME_2[which(rn$NAME_2=="Groaíras")]<-"Grossos"
```

Foi também detectado que o dataframe de polígonos ainda inclui os municípios de "Fernando de Noronha" – hoje pertencente ao estado de Pernambuco – e "Poço Dantas" – pertencente ao estado da Paraíba. Assim, estes polígonos precisam ser removidos! Mas vamos deixar isso por sua conta!

Após estas correções, estamos prontos para aplicar uma construção atípica (pois devemos dizer como campos com nomes distintos devem ser relacionados) da função ***merge***.

```
rn <- merge(x=rn, y=base, by.x="NAME_2", by.y="cidade") # Faz um merge dos dataframes
```

Agora que temos todas as informações que precisamos para plotar o gráfico do RN de acordo com a sua população, precisamos definir alguns parâmetros para a visualização. Primeiro iremos estabelecer intervalos para caracterizar a densidade populacional em cada município. No R, a criação de intervalos pode ser realizada através do uso da função ***cut()***. Podemos então usar estes intervalos para representar a densidade populacional de acordo com uma cor específica. Esta informação é adicionada ao dataframe como uma nova coluna (***col_no***). Essa coluna servirá como referência para determinar a cor de cada munícipio no mapa a ser plotado.

```
# Criando os intervalos e classificando
col_no = as.factor(cut(rn$pop2010, breaks =
c(0,3000,10000,100000,300000,500000,800000,1000000), labels=c("<3k", "3k-10k", "10k-
100k","100k-300k", "300k-500k", "500k-800k", ">800k"), right= FALSE))

# Nomeando os intervalos – irá aparecer na legenda do grafico
levels(col_no) = c("<3k", "3k-10k", "10k-100k","100k-300k", "300k-500k", "500k-800k",
">800k")

# Adicionando a informação da categoria no dataframe
rn$col_no = col_no
```

Paletas são fundamentais para oferecer uma boa representação daquilo que ser quer comunicar. Podemos acessar a paleta de cores de várias formas no R, utilizando o pacote base ou uma série de outros pacotes. Em nosso exemplo, utilizaremos a palheta *Brewer* com tons de verde. Existem muitas outras palhetas disponíveis. A escolha da palheta não é apenas uma questão de beleza, mas deve ter em conta a melhor visibilidade das características que desejamos evidenciar no mapa.

```
library(RColorBrewer)

myPalette = brewer.pal(7,"Greens")
```

Aqui já é possível plotar o mapa com a representação da população de cada município na forma de cores distintas.

```
spplot(rn, "col_no", col=grey(.9), col.regions=myPalette, main="Municípios do RN")
```

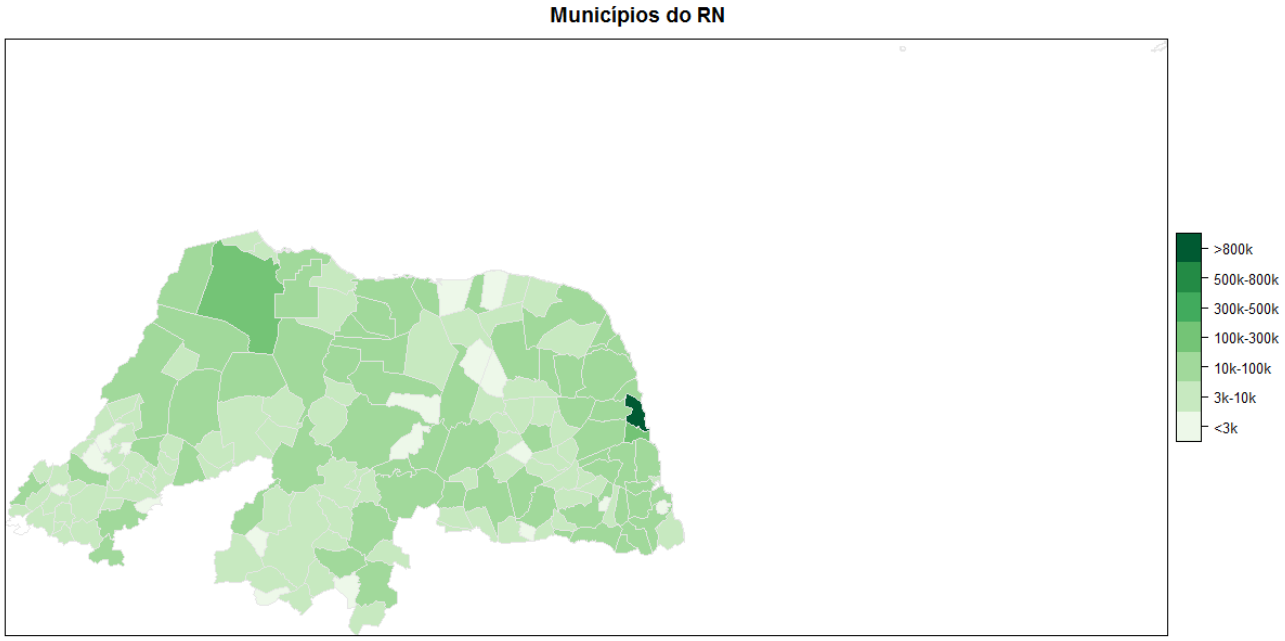

Figura 5. Mapa do RN com a representação da densidade populacional em cores.

Podemos melhorar a visualização ao inserir rótulos ou marcar pontos específicos no mapa. Para efeito de exemplo, imagine que queiramos indicar no mapa as cidades de Natal, Mossoró e Pau dos Ferros. Queremos ainda rotular o nome de cada cidade no mapa.

Deve ficar claro que a indicação de qualquer ponto em um mapa georeferenciado deve ser feito também através de uma georeferência ou, em outras palavras, precisamos indicar precisamente as coordenadas destas cidades no mapa. Para isso, usamos a biblioteca ***ggmap***, que inclui a função ***geocode()***. Seu uso é bem simples: basta informar um endereço alvo e recebemos de volta sua latitude e longitude. Tecnicamente, esta função acessa a API do Google Maps e faz uma pesquisa pelo endereço informado. Obviamente, nem todos os endereços são unívocos e, com isso, devemos "ajudar" a encontrar o endereço que queremos. Por essa razão, adicionaremos sempre "Brasil+RN" na busca pelos nomes de municípios.

Vale ressaltar que, embora neste tutorial o ***ggmap*** seja usado apenas para obter apenas a geolocalização dos pontos que queremos plotar no mapa, esta biblioteca inclui diversas funções interessantes, permitindo carregar e plotar mapas utilizando diversos serviços, como Google Maps, OpenStreetMap, Stamen Maps e CloudMade. Ainda, é possível solicitar diferentes estilos como satélite, elevação, híbrido ou ruas.

Segue o trecho do script que usamos para obter as geolocalizações para os pontos que queremos indicar no mapa.

```
library(ggmap) # importa a biblioteca ggmap para obter a geolocalização

# Define o nome das cidade-alvos – a ser apresentada como label no mapa
nomes = c("Natal","Mossoró","Pau dos Ferros")

# Define or argumentos de busca para as cidade-alvos – a ser usada pelo ggmap
nam = c("Natal+Brazil+RN","Mossoro+Brazil+RN","Pau dos Ferros+Brazil+RN")

# Busca a geolocalização (Google) para cada cidade
pos = geocode(nam)

# Define a posição dos labels como sendo um pouco acima dos pontos
tlat = pos$lat+0.05    # -- the city name will be above the marker

# Cria um daframe com as informações (nome da cidade, longitude, latitude e posição do
label)
cities = data.frame(nomes, pos$lon,pos$lat,tlat)

# Nomeia as colunas de longitude e latitude
names(cities)[2] = "lon"
names(cities)[3] = "lat"
```

Agora é possível criar os elementos de marcação e texto (label) a serem plotados no gráfico.

```
# Criando os labels
text1 = list("panel.text", cities$lon, cities$tlat, cities$nomes,col="black", cex = 0.5)

# Criando os apontamentos
mark1 = list("panel.points", cities$lon, cities$lat, col="blue")
```

Agora é possível plotar o mapa contendo todas as informações.

```
spplot(rn, "col_no",
       sp.layout=list(text1,mark1),
       main="Municípios RN",
       col=grey(.9), col.regions=myPalette)
```

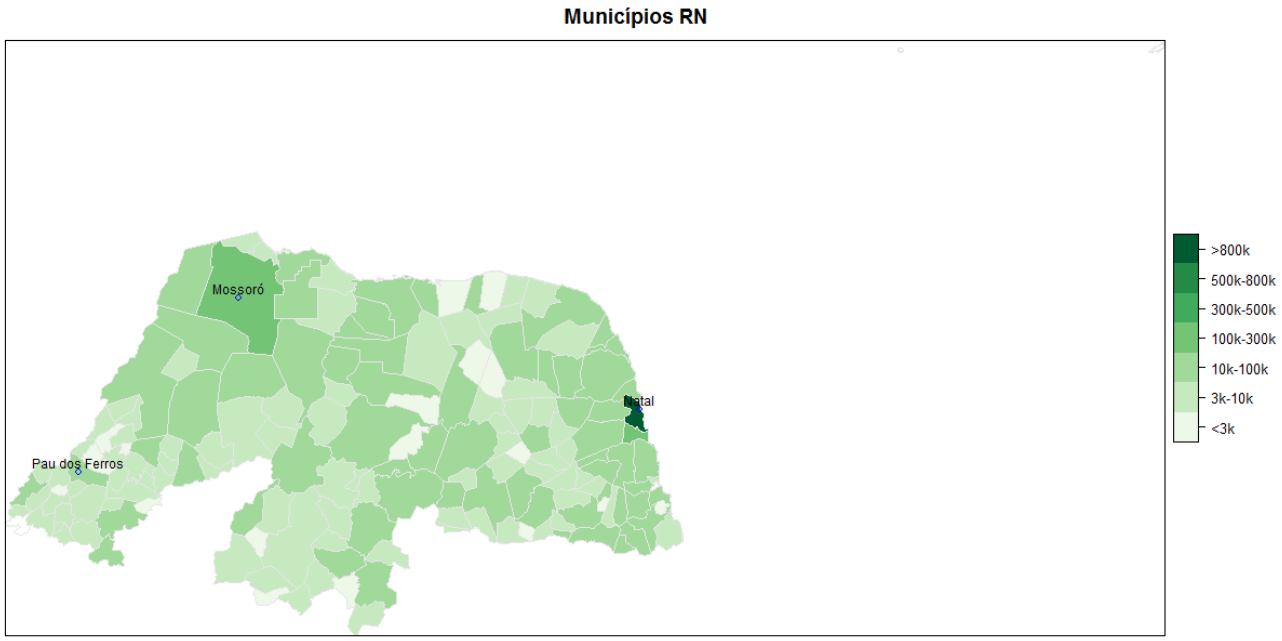

Figura 6. Mapa do RN com a representação da densidade populacional em cores e com apontamento e labels para as cidades de Natal, Mossoró e Pau dos Ferros.

Caso queiramos evidenciar ainda mais as características da população, podemos alterar a palheta de cores utilizada e plotar novamente o gráfico, como mostrado a seguir.

```
myPalette = rainbow(7)  # Alterando a palheta de cores

spplot(rn, "col_no",
       sp.layout=list(text1,mark1),
       main="Municípios RN",
       col=grey(.9), col.regions=myPalette)
```

Isso irá produzir o gráfico a seguir.

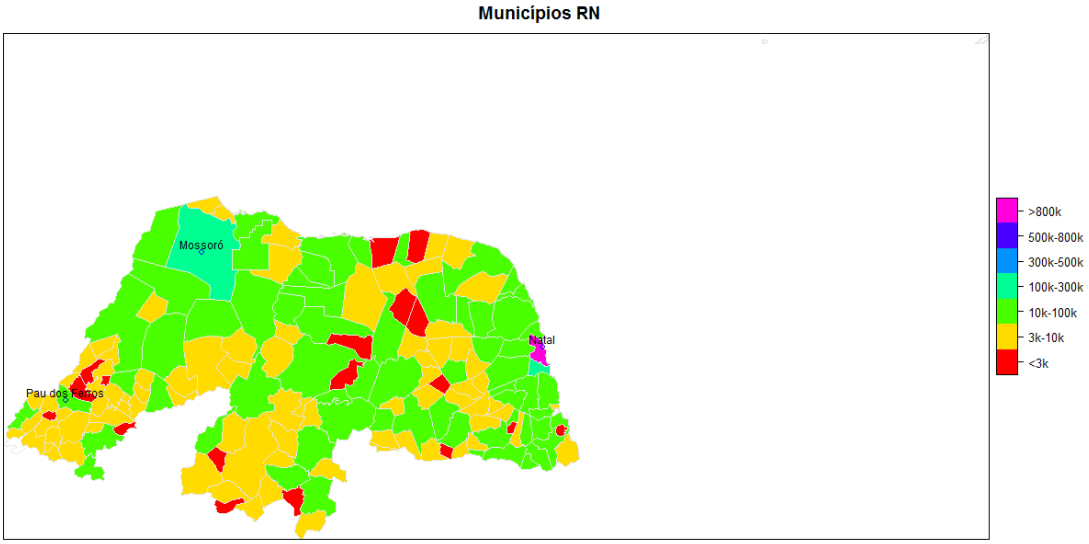

Figura 7. Gráfico após a alteração da palheta de cores.

Já que falamos em mapas, vamos dar um gostinho do uso do GoogleMaps!!! Como dito anteriormente, podemos fazer uso de outras funções da biblioteca ***ggmap*** para isso. Vamos tomar como exemplo simples, gerar o mapa de Natal.

Carregamos o mapa pelo Google Maps com o estilo de ruas para a geolocalização de Natal. O argumento *zoom* é um número inteiro entre 3 e 21 com valor padrão de 10, onde 3 representa o zoom do continente, 21 para prédios e 10 a cidade. O argumento *maptype* indica o tipo de renderização do mapa, podendo ser "terrain", "terrain-background", "satellite", "roadmap" ou "hybrid". Segue o trecho do script em R.

```
library(ggmap) # importa a biblioteca ggmap para obter a geolocalização

natalMap = get_map(location = "Natal+RN+Brasil", zoom = 11,
             source = "google", maptype="roadmap")

ggmap(natalMap)
```

Isso irá produzir o gráfico a seguir.

Figura 8. Exemplo de mapa de Natal do Google Maps gerado pelo R com a biblioteca ggmap.

Considerações Finais

Nesta atividade apresentamos uma breve introdução à *webscraping* e manipulação de mapas em R. Estas técnicas complementam o conteúdo já abordado na disciplina e permite dar uma visão geral das potencialidades do R na manipulação, integração, análise e visualização de dados.

Além do foco na linguagem R, a principal lição que deve ser ressaltada é que na área da Saúde são inúmeras e diversas as fontes de dados. Por isso, precisamos nos capacitar com o uso de diferentes ferramentas para poder acessar e consumir estes dados, permitindo complementar os estudos realizados na área.

Esperamos que o (breve) conteúdo apresentado aqui sirva de motivação para estudos mais aprofundados nas ferramentas abordadas ou mencionadas aqui, bem como na busca por outras soluções.

ATIVIDADE

Questão 01. Se você é um bom cientista de dados, observou que o dataframe de polígonos obtido no repositório de base de dados espaciais "Global Administrative Areas" ainda registra "Fernando de Noronha" e "Poço Dantas" como parte do RN. Corrija este detalhe, removendo este município de seu dataframe e volte a plotar os gráficos. Como uma tarefa extra (isso exigirá uma pesquisa simples sobre as bibliotecas do R), salve cada gráfico em formato de imagem (PNG, JPEG, etc).

Questão 02. Seguindo a mesma idéia apresentada neste tutorial, apresente uma visualização espacial (usando mapa) de algum indicador de saúde para o RN, obtido da plataforma TabNet. Exporte em formato CSV e faça a integração com o mapa. Justifique a escolha do indicador.

Questão 03. Como uma melhoria da questão anterior, apresente o mapa com as cores representando a densidade populacional e utilize o rótulo (*label*) para apresentar as informações do indicador de saúde escolhido, mas apenas para os municípios de "Natal", "Caiocó", "Mossoró", "Pau dos Ferros" e "Currais Novos".