Introdução à Linguagem R

Encontro 4

Davi Moreira

21 de Maio, 2019

Sumário

1 Encontro 4

1.1 Estrutura do encontro

- Regressão linear: aplicação e visualização;
- for, while e nossas próprias funções;
- purrr package;
- Web Scraping;
- Mapas;

Até o final do encontro o aluno deverá ser capaz de:

- Produzir gráficos que permitam análise dos resultados de modelos preditivos;
- criar suas próprias funções;
- requisitar dados da web de forma automatizada;
- Georreferenciar dados;

2 Regressão linear

A equação linear apresenta como principais características: - O coeficiente angular a da reta é dado pela tangente da reta; - A cota da reta em determinado ponto é o coeficiente linear denominado b que é o valor de y quando x for igual a zero.

Possui a seguinte fórmula:

$$y = ax + b + \epsilon$$

onde:

- x é a variável independente ou preditora;
- y é a variável dependente ou predita;
- ϵ é chamado de erro que corresponde ao desvio entre o valor real e o aproximado (pela reta) de y. Isso porque sempre há observações amostrais que não são pontos da reta.

A equação linear pode ser obtida no `R` por meio da função `lm()` que serve para calcular a regressão linear simples.

```
# construindo variavel dependente

censo_pnud_pe_sel$docentes_esc <- censo_pnud_pe_sel$n_docentes /
                                    censo_pnud_pe_sel$n_escolas

reg <- lm(IDHM_E ~ docentes_esc + n_matriculas + escolas_energia_inex,
          data = censo_pnud_pe_sel)

names(reg)

# Os mais importantes listados são os seguintes:

# regressão$fitted.values ou predict(), que calcula os valores preditos da variável
# resposta para cada elemento da amostra (faz uma previsão);

# regressão$residuals: calcula o erro ou os resíduos (valor observado - valor predito)
# para cada ponto da amostra;
# regressão$coefficients:  obtém uma estimativa dos coeficientes da regressão
```

```r
options(scipen=999)
summary(reg)
```

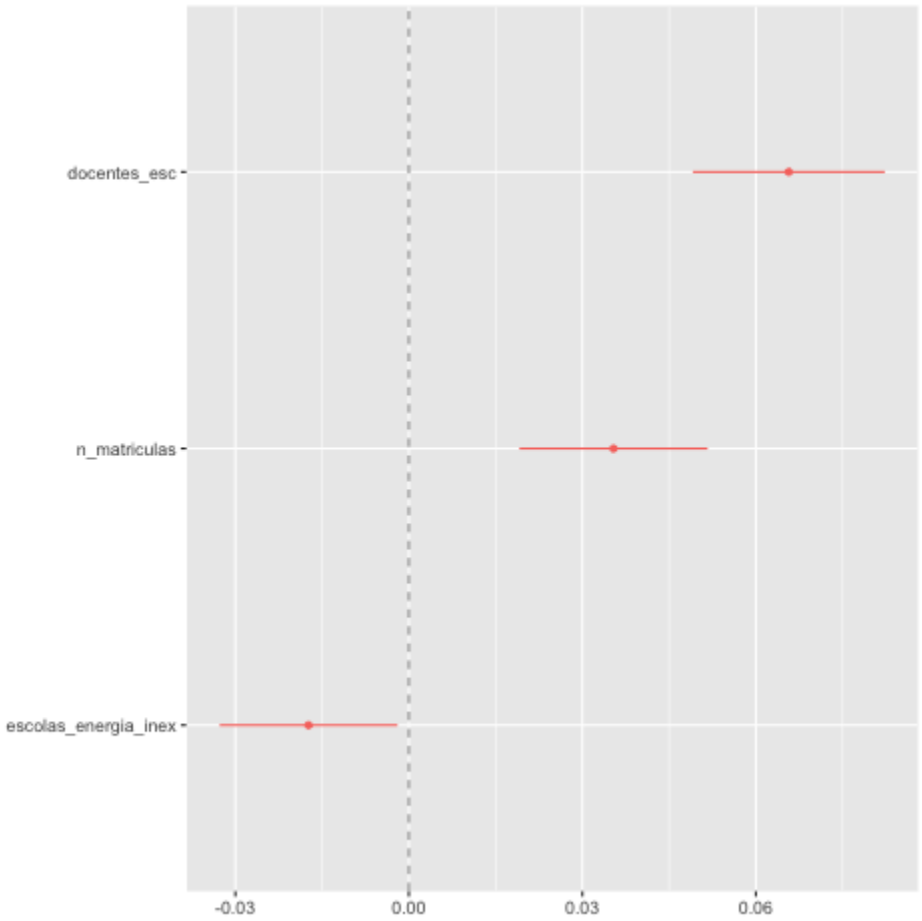
2.1 Comunicando resultados do modelo de regressão

Vimos que a comunicação da análise de dados é uma etapa importante da atividade científica. Para torná-la eficiente e de fácil interpretação, nesse curso priorizamos a visualização gráfica dos dados ao invés de tabelas. Para tanto, nos baseamos no artigo *Using graphs instead of tables in political science* de Kastellec e Leoni (2007).

```r
if(require(dotwhisker) == F) install.packages('dotwhisker'); require(dotwhisker)
if(require(broom) == F) install.packages('broom'); require(broom)

censo_pnud_pe_sel$docentes_esc <- censo_pnud_pe_sel$n_docentes /
  censo_pnud_pe_sel$n_escolas

reg <- lm(IDHM_E ~ docentes_esc + n_matriculas + escolas_energia_inex,
          data = censo_pnud_pe_sel)
summary(reg)

dwplot(reg,  vline = geom_vline(xintercept = 0, colour = "grey60", linetype = 2))

setwd("./imagens")
png(filename="modelo_idhe.png")
dwplot(reg,  vline = geom_vline(xintercept = 0, colour = "grey60", linetype = 2))
dev.off()
```

```
## Loading required package: png
```

```
## Loading required package: grid
```



2.2 Links úteis:

- Análise exploratória
- Regressão Linear
- Pacote ggplot2
- Data Visualisation
- CursoR: ggplot2

3 `for`, `while` e nossas próprias funções

As funções `for()` e `while()` implementam o controle de fluxo no `R`. A escolha de qual usar vai depender do contexto e objetivo do código.

3.1 `for`

```
# for()

for (i in 1:10) {
```

```
  print (i)
}
```

3.2 while

```
# while ()
x = 1

while (x <= 10 )  {
  print (x)
  x = x + 1
}
```

3.3 Criando funções

```
soma_dois <- function(x) { x + 2 }

soma_dois(4)

obj <- 1:15

soma_dois(obj)
```

4 purrr package

Para uma boa introdução sobre o pacote, veja o seguinte material:

- Curso R - Purrr
- Happy R Users Purrr – Tutorial
- Purrr Tutorial

```
install.packages("tidyverse")
library(purrr)
```

```
soma_dois <- function(x) { x + 2 }
obj <- 1:15

obj <- map(obj, soma_dois)
obj
```

```r
# como a funcao map retorna uma lista, podemos usar sufixos para retornar um tipo
# de vetor específico

map_dbl(obj, soma_dois)
```

5 Web Scraping

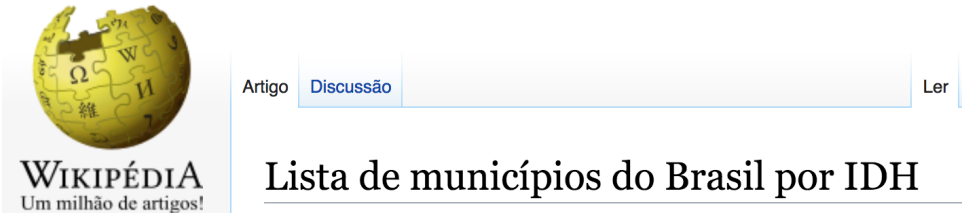
Web Scraping é uma técnica de extração de dados utilizada para coletar conteúdo publicado na internet por meio de procedimentos automatizados.

5.1 Tipos de conteúdo disponível

5.1.1 Código fonte

É possível conhecer o código fonte de um site ao clicar com o botão direito do mouse no conteúdo da página.

- Wikipedia

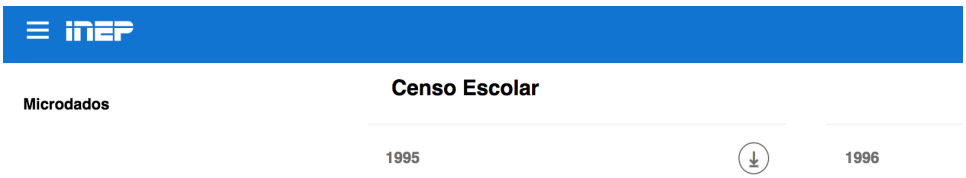


- Deputados

5.1.2 Arquivos para download

Além do conteúdo diretamente publicado na página, pode ser de interesse fazer o download de arquivos disponíveis.

- Censo Escolar



- TCE

5.1.3 Web services

Os Web services são componentes que permitem às aplicações enviar e receber dados. Um dos motivos que tornam os Web Services atrativos é o fato deste modelo ser baseado em tecnologias standards, em particular XML e HTTP (Hypertext Transfer Protocol). Os Web Services são utilizados para disponibilizar serviços interativos na Web, podendo ser acessados por outras aplicações. O objetivo dos Web Services é a comunicação de aplicações através da Internet.

- Web service da Câmara dos Deputados

5.2 Pacotes para raspagem de dados

Há diversos pacotes para raspagem de dados com o `R`. Abaixo segue um lista com os principais. Para referências sobre seu uso, consulte os links indicados, este tutorial sobre o 'rvest' e este capitulo sobre web scraping.

- 'httr'
- 'xml2'
- 'rvest'

Como o site Curso-R destaca, esses pacotes não são suficientes para acessar todo tipo de conteúdo da web. Páginas com conteúdo produzido na linguagem `javascript`, por exemplo, precisam de outras ferramentas para acesso a seu conteúdo. Nesses casos, é necessário "simular" um navegador que acessa a página web e realiza consultas. Uma das melhores ferramentas para isso é o selenium, abaixo indicado.

- 'RSelenium'

5.3 Obtendo conteúdo

5.4 Etapas para raspagem de dados na web

1. Conhecer detalhadamente o caminho para acesso aos dados
2. Armazenar todos os caminhos de acesso aos dados de forma amigável ao programa
3. Obter os dados

4. Processar os dados obtidos

5.5 Conteúdo de páginas

5.5.1 Código Fonte:

Podemos facilmente obter o código fonte de um endereço na internet com o uso da função `readLines`.

```
if(require(tidyverse) == F) install.packages('tidyverse'); require(tidyverse)
if(require(rvest) == F) install.packages('rvest'); require(rvest)
if(require(httr) == F) install.packages('httr'); require(httr)
if(require(xml2) == F) install.packages('xml2'); require(xml2)

## Etapas para raspagem de dados na web ----

# 1. Conhecer detalhadamente o caminho para acesso aos dados
# 2. Armazenar todos os caminhos de acesso aos dados de forma amigável ao programa
# 3. Obter os dados
# 4. Processar os dados obtidos

## Conteúdo de páginas ----

### **Código Fonte:** ----

if(require(tidyverse) == F) install.packages('tidyverse'); require(tidyverse)
if(require(rvest) == F) install.packages('rvest'); require(rvest)
if(require(httr) == F) install.packages('httr'); require(httr)
if(require(xml2) == F) install.packages('xml2'); require(xml2)

################################################################################
# ETAPA 1. Conhecer detalhadamente o caminho para acesso aos dados
# ETAPA 2. Armazenar todos os caminhos de acesso aos dados de forma amigável ao programa

# defnindo endereço da web
link <- "https://pt.wikipedia.org/wiki/Lista_de_munic%C3%ADpios_do_Brasil_por_IDH"

################################################################################
# ETAPA 3. Obter os dados
conteudo <- readLines(link)  # obtem o codigo fonte
head(conteudo)

# Vamos verificar a posição de Fenando de Noronha no vetor 'conteudo'.
grep("Fernando", conteudo)
conteudo[782]  # linha com o IDH de Fernando de Noronha
conteudo[782 + 4]  # linha com o IDH de Fernando de Noronha

conteudo[782 + 9]  # Próximo município
conteudo[782 + 9 + 9] # Parece haver um padrão

# Com o objeto 'conteudo' já seria possível obter os dados para criacão do data frame
# com o IDH dos municípios.

################################################################################
```

```r
# ETAPA 4. Processar os dados obtidos
# vamos selecionar todas linhas que apresentem os nomes dos municipios

grep("São Caetano", conteudo)  #
grep("Santa Maria", conteudo)  #

indice <- 107
nomes_munic <- NULL
i <- 1

while(indice < 1083){
  if(i==1){
    nomes_munic[i] <- conteudo[indice]
  } else{
    nomes_munic[i] <- conteudo[indice+9]
  }
  indice <- indice + 9
  i <- i + 1
}

nomes_munic

?regex

nomes_munic <- gsub("[[:print:]]+\">", "", nomes_munic)
nomes_munic <- gsub("</a>", "", nomes_munic)
nomes_munic <- gsub("</b>", "", nomes_munic)
nomes_munic <- gsub("<b>", "", nomes_munic)

nomes_munic

# Poderíamos realizar procedimento semelhante para obter os IDHs municipais, os
# nomes da UFs e assim construir nosso data.frame
```

5.5.1.1 Atividade prática:

```r
# Identifique em qual linha do vetor 'conteudo' está Pernambuco. Adapte o exemplo visto
# para obter um vetor com os nomes das UFs.
```

5.5.2 Obtendo tabelas em html:

```r
################################################################################
# ETAPA 1. Conhecer detalhadamente o caminho para acesso aos dados
# ETAPA 2. Armazenar todos os caminhos de acesso aos dados de forma amigável ao programa
link <- "https://pt.wikipedia.org/wiki/Lista_de_munic%C3%ADpios_do_Brasil_por_IDH"


################################################################################
# ETAPA 3. Obter os dados
# ETAPA 4. Processar os dados obtidos

bd <- link %>%
```

```
  httr::GET() %>%
  xml2::read_html() %>%
  rvest::html_node('table') %>%
  rvest::html_table(header = TRUE)

bd
class(bd)

# E quando o link possui mais de uma tabela?

################################################################################
# ETAPA 1. Conhecer detalhadamente o caminho para acesso aos dados
# ETAPA 2. Armazenar todos os caminhos de acesso aos dados de forma amigável ao programa
link <- "https://pt.wikipedia.org/wiki/Lista_de_campe%C3%B5es_do_futebol_brasileiro"

################################################################################
# ETAPA 3. Obter os dados
bd <- link %>%
  httr::GET() %>%
  xml2::read_html() %>%
  rvest::html_nodes('table') %>%  # veja que utilizamos outra função
  rvest::html_table(header = TRUE)

class(bd)

################################################################################
# ETAPA 4. Processar os dados obtidos
bd1 <- bd[[1]]
bd2 <- bd[[2]]
bd3 <- bd[[3]]
```

5.5.2.1 Atividade prática:

```
# Com o link abaixo, desenvolva um progrma que obtenha o endereço das páginas de todos
# os deputados federais da atual legislatura alocando-os num vetor.

link <- "http://www.camara.leg.br/internet/deputado/DepNovos_Lista.asp?
          Legislatura=54&Partido=QQ&SX=QQ&Todos=None&UF=QQ&condic=QQ&forma=lista&
          nome=&ordem=nome&origem="
```

5.6 Download de arquivos

```
################################################################################
# ETAPA 1. Conhecer detalhadamente o caminho para acesso aos dados
link <- "https://www.tce.pe.gov.br/internet/index.php/relatorios-de-gestao-fiscal-2"

################################################################################
# ETAPA 2. Armazenar todos os caminhos de acesso aos dados de forma amigável ao programa
link_relatorios <- link %>% read_html %>% html_nodes("a") %>% html_attr('href')
link_relatorios <- link_relatorios[grep("rdg", link_relatorios)]

################################################################################
```

```r
# ETAPA 3. Obter os dados
mainDir <- paste(getwd(), "/dados/", sep = "")
subDir <- "relatorios_tce"

dir.create(file.path(mainDir, subDir), showWarnings = FALSE)

setwd("./dados/relatorios_tce/")

for( i in 1:length(link_relatorios)){
   download.file(link_relatorios[i], paste0(as.character(c(2017:2006))[i], ".pdf"), mode="auto")
}

download.file(link_relatorios)

################################################################################
# ETAPA 4. Processar os dados obtidos
if(require(pdftools) == F) install.packages('pdftools'); require(pdftools)

setwd("./relatorios_tce/")
rdg2017 <- pdf_text("2017.pdf")

length(rdg2017)
head(rdg2017)

# Preparando data.frame para nuvem de palavras
# Instalando pacotes
if(require(tm) == F) install.packages('tm'); require(tm)
if(require(SnowballC) == F) install.packages('SnowballC'); require(SnowballC)
if(require(wordcloud) == F) install.packages('wordcloud'); require(wordcloud)
if(require(RColorBrewer) == F) install.packages('RColorBrewer'); require(RColorBrewer)

docs <- Corpus(VectorSource(rdg2017))

# convertendo o texto em caixa baixa
docs <- tm_map(docs, content_transformer(tolower))

# removendo números
docs <- tm_map(docs, removeNumbers)

# removendo stopwords (artigos, preposições, etc.)
docs <- tm_map(docs, removeWords, stopwords("portuguese"))

# removendo pontuação
docs <- tm_map(docs, removePunctuation)

# eliminando espaços
docs <- tm_map(docs, stripWhitespace)

# stemming
docs <- tm_map(docs, stemDocument, language = "portuguese")

# document term matrix - matriz de documentos e termos
dtm <- TermDocumentMatrix(docs)
```

```r
# criando data.frame
m <- as.matrix(dtm)
v <- sort(rowSums(m), decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)
head(d)
d <- d[-3,]  # removendo caracter especial
head(d)

# nuvem de palavras
set.seed(1234)
wordcloud(words = d$word, freq = d$freq, min.freq = 1,
          max.words=100, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(8, "Dark2"))
```

5.6.0.1 Atividade prática:

```r
# Crie um novo diretório e desenvolva um programa que faça o download dos arquivos
# em formato .zip na página de dados do Censo Escolar (http://inep.gov.br/microdados).
# Para otimizar nosso tempo, selecione apenas os microdados do Censo Escolar de
# 1996, 2006, 2016.
```

5.7 Web service

```r
if(require(httr) == F) install.packages('httr'); require(httr);
if(require(XML) == F) install.packages('XML'); require(XML);
if(require(xml2) == F) install.packages('xml2'); require(xml2);

##############################################################################
# ETAPA 1. Conhecer detalhadamente o caminho para acesso aos dados
# ETAPA 2. Armazenar todos os caminhos de acesso aos dados de forma amigável ao programa
```

```r
link <- paste0("http://www.camara.leg.br/SitCamaraWS/Deputados.asmx/ObterDeputados")

##############################################################################################
# ETAPA 3. Obter os dados
response <- GET(link)

##############################################################################################
# ETAPA 4. Processar os dados obtidos
data <- xmlParse(response, encoding = "UTF-8")
ls <- xmlToList(data)

names(ls$deputado)

ideCadastro <- NULL
condicao <- NULL
matricula <- NULL
idParlamentar <- NULL
nome <- NULL
nomeParlamentar <- NULL
urlFoto <- NULL
sexo <- NULL
uf <- NULL
partido <- NULL
email <- NULL

for(i in 1:length(ls)){
  ideCadastro[i] <- ls[[i]]$ideCadastro
  condicao[i] <- ls[[i]]$condicao
  matricula[i] <- ls[[i]]$matricula
  idParlamentar[i] <- ls[[i]]$idParlamentar
  nome[i] <- ls[[i]]$nome
  nomeParlamentar[i] <- ls[[i]]$nomeParlamentar
  urlFoto[i] <- ls[[i]]$urlFoto
  sexo[i] <- ls[[i]]$sexo
  uf[i] <- ls[[i]]$uf
  partido[i] <- ls[[i]]$partido
  email[i] <- ls[[i]]$email
}

bd <- data.frame(ideCadastro, condicao, matricula, idParlamentar, nome,
                 nomeParlamentar, urlFoto, sexo, uf, partido, email)

head(bd)
```

5.7.0.1 Atividade prática:

Com a base de dados obtida, utilize a variável ideCadastro para obter detalhes dos Deputados Federais que representam o Estado de Pernambuco, confome permitido pelo link ObterDetalhesDeputado

5.7.0.2 Atividade Prática

```r
# Utilizando o vetor de endereços das páginas dos deputados federais obtido com o link abaixo,

link <- "http://www.camara.leg.br/internet/deputado/DepNovos_Lista.asp?
        Legislatura=54&Partido=QQ&SX=QQ&Todos=None&UF=QQ&condic=QQ&forma=lista&
        nome=&ordem=nome&origem="

# Acesse cada uma das páginas e monte uma base de dados que tenha as seguintes
# informações biográficas de cada deputado: Nome, Data de Nascimento, Naturalidade,
# Profissao, Filiacao e Escolaridade.

# Qual a proporção de Deputados Federais com ensino superior?
```

5.7.0.3 Referência adicional:

- Webscraping with 'Python'.

6 Mapas

O IBGE divulga em sua página bases cartográficas do páis em diferentes níveis. Também chamados de `shapefiles` estes arquivos serão utilizados para produção de mapas no `R` podem ser encontrados nos links abaixo

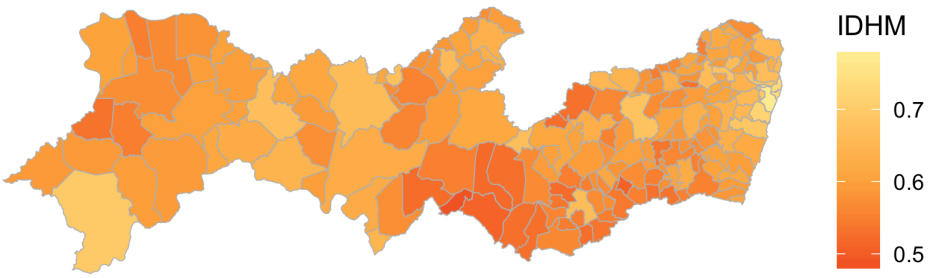
- https://mapas.ibge.gov.br/bases-e-referenciais/bases-cartograficas/malhas-digitais
- ftp://geoftp.ibge.gov.br/organizacao_do_territorio/malhas_territoriais/malhas_municipais/municipio_2015/UFs/PE/

6.1 Mapas com o `ggplot2`

```r
# pacotes ------------------------------------------------------------------
if(require(rgdal) == F) install.packages("rgdal"); require(rgdal)
if(require(maptools) == F) install.packages("maptools"); require(maptools)
if(require(ggmap) == F) install.packages("ggmap"); require(ggmap)
if(require(mapproj) == F) install.packages("mapproj"); require(mapproj)

if(require(ggplot2) == F) install.packages("ggplot2"); require(ggplot2)
if(require(tidyverse) == F) install.packages("tidyverse"); require(tidyverse)

# carregando bases ---------------------------------------------------------

# Carregando shapefile
shapefile_pe <- readOGR("./pe_municipios/", "26MUE250GC_SIR")

plot(shapefile_pe)

shapefile_pe@data

# Convertendo o shapefile para dataframe ----
shapefile_df <- fortify(shapefile_pe)

dim(shapefile_df)
```

```r
names(shapefile_df)
head(shapefile_df)

shapefile_data <- fortify(shapefile_pe@data)
shapefile_data$id <- row.names(shapefile_data)

shapefile_df <- full_join(shapefile_df, shapefile_data, by="id")

names(shapefile_df)
head(shapefile_df)

# Agora vamos remover Fernando de Noronha (2605459) da base e produzir o mapa novamente ----

shapefile_df <- shapefile_df %>% filter(CD_GEOCMU != "2605459")

# mapa ggplot
map <- ggplot() +
  geom_polygon(data = shapefile_df,
               aes(x = long, y = lat, group = group, fill = IDHM),
               colour = "black", fill = 'white', size = .2) +
  coord_map()

map

# Fazendo união com a base do CensoEscolar+PNUD ----
censo_pnud_pe_sel$Codmun7 <- as.factor(censo_pnud_pe_sel$Codmun7)
shapefile_df <- shapefile_df %>% left_join(censo_pnud_pe_sel,
                                           by = c("CD_GEOCMU" = as.character("Codmun7")))
head(shapefile_df)

# sugestão para escolha de cores ----
# http://colorbrewer2.org/#type=sequential&scheme=BuGn&n=3
# https://www.w3schools.com/colors/colors_picker.asp
# https://ggplot2.tidyverse.org/reference/scale_gradient.html
# https://www.w3schools.com/colors/colors_picker.asp

# mapa IDHM ----
map <- ggplot() + geom_polygon(data = shapefile_df,
                               aes(x = long, y = lat, group = group, fill = IDHM),
                               colour = "gray") +
  theme_void() +
  coord_map()
map


# mapa IDHM - fundo vazio ----
map <- ggplot() + geom_polygon(data = shapefile_df,
                               aes(x = long, y = lat, group = group, fill = IDHM),
                               colour = "gray", size = .2) +
  theme_void() +  # essa é a função que dexa o fundo vazio
  coord_map()
map
```

```r
# mapa IDHM - fundo vazio e nova cor da escala ----

map <- ggplot() + geom_polygon(data = shapefile_df,
                               aes(x = long, y = lat, group = group, fill = IDHM),
                               colour = "gray", size = .2) +
  theme_void() +  # essa é a função que dexa o fundo vazio
#  scale_fill_manual(values = c("Black", "Orange", "Brown")) +
  scale_fill_gradient2(low = "#f03b20", mid="#feb24c", high = "#ffeda0",
                       midpoint = median(shapefile_df$IDHM),
                       limits = range(shapefile_df$IDHM)) +
  coord_map()

map

setwd("./imagens")
ggsave(filename = "pernambuco_municipios.png", map)
```



6.1.1 Atividade prática

1. Faça um mapa apenas com os municípios da Regiao Metropolitana do Recife e seu IDHM. Use o link abaixo para auxiliar nessa tarefa:

- https://pt.wikipedia.org/wiki/Regi%C3%A3o_Metropolitana_do_Recife

6.2 Mapas com o `ggplot2` e o Google Maps

Recentemente o Google Maps passou a exigir de todos os desenvolvedores uma API específica para uso gratuito do serviço.

- Para cadastrar um projeto e fazer obter uma API, utilize este link.
- Para informações sobre esta mudança, veja:
- GitHub do pacote ggmap.
- GitHub do pacote ggmap - Issue 51.
- Questão 1 StackOverflow.
- Questão 2 StackOverflow.

Como é possível perceber pelas informações dos links acima, adaptações no pacote ggmap estão em desenvolvimento e, em breve, devem compor a versão oficial do pacote para uso gratuito do serviço. De todo modo, pode-se fazer uso dos avanços em desenvolvimento com o código abaixo[1].

```
# instalando pacote direto do repositório Git
if(!requireNamespace("devtools")) install.packages("devtools")
devtools::install_github("dkahle/ggmap", ref = "tidyup")

# carregando pacote
library(ggmap)

# registrando api
register_google(key = "sua_api_aqui")

# mapas de Recife
recife1 <- get_map("Recife")
ggmap(recife1)

recife1 <- get_map("Recife", maptype = c("satellite"))
ggmap(recife1)

recife2 <- get_map("Recife", zoom = 10)
ggmap(recife2)

recife3 <- get_map("Recife", zoom = 12)
ggmap(recife3)

recife4 <- get_map("Av. Acadêmico Hélio Ramos - Cidade Universitária, Recife - PE,
                   50670-901", zoom = 15)
ggmap(recife4)
```

[1] Dada a intensa e resistente comunidade desenvolvedora e colaborativa do R, pode-se ter a certeza de que uma opção gratuita sempre estará disponível para uso.

```r
cfch <- geocode("CFCH - Cidade Universitária, Recife - PE")
cfch

ggmap(recife4) + geom_point(data = cfch, aes(lon, lat), color = "red", size = 2)

reitoria_ufpe <- geocode("Reitoria UFPE - Cidade Universitária, Recife - PE")

mapa_ufpe <- ggmap(recife4) + geom_point(data = cfch, aes(lon, lat),
                                         color = "red", size = 2) +
  geom_point(data = reitoria_ufpe, aes(lon, lat), color = "blue", size = 2)

mapa_ufpe

setwd("./imagens")
ggsave(filename = "mapa_ufpe.png", mapa_ufpe)
```

6.2.1 Atividade prática:

- Use o mapa `recife3` para apresentar a imagem do google maps e adicione as fronteiras da cidade de Recife.
- Obtenha um mapa do Google Maps com zoom num endereço qualquer de sua escolha e apresente um mapa com um ponto azul na localização do endereço.

7 O que não vimos no curso

- Automatização de relatórios com o R Markdown.
- Processamento de linguagem natural.
- Métodos de aprendizagem computacional:
  - Análise automatizada de conteúdo com o `R`.
  - Análise automatizada de imagens com o `R`.