# Web Scraping

## Páginas dinâmicas

# Selenium

# O que é Selenium?

- Selenium é uma ferramenta que permite **automatizar um navegador**!
- Suporta alguns *backends* diferentes: PhantomJS, Firefox, Chrome, etc.
- Diferentemente do web scraping normal, não precisamos nos preocupar com nenhuma requisição HTTP
    - O Selenium literalmente cria um navegador invisível para o qual você pode passar as **ações** a serem tomadas
    - Por ser uma sessão interativa, não há dificuldades em exibir conteúdo dinâmico
    - Não é necessário compreender o *networking* do site: tudo é *headless*

# Por que não usá-lo sempre?

- Vantagens:
  - Fácil de entender
  - Permite raspar dados dinâmicos
  - Permite *screen shots*
- Desvantagens:
  - Lento e de difícil paralelização
  - Bastante sensível
  - `RSelenium` tem fama de ser **quebrado**

# WebDriver

- Não existe uma diferença real entre "Selenium" e "WebDriver"
    - O nome correto da ferramenta é Selenium WebDriver
- A diferença está no R: pacotes `RSelenium` e `webdriver`
    - `RSelenium` exige certo conhecimento para fazer funcionar
    - `webdriver` foi feito pela própria RStudio para resolver o problema
- O `webdriver` funciona somente com o PhantomJS, mas isso não é necessariamente um problema
- Instalar é fácil, fazer funcionar é mais ainda

# PhantomJS

- O PhantomJS é um navegador *headless* baseado em JavaScript feito especificamente para interação automatizada com páginas da web

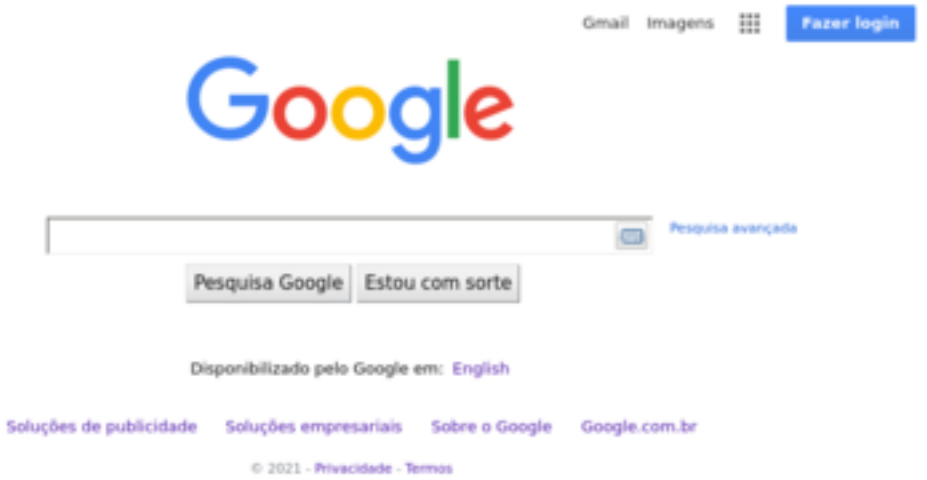
```
library(webdriver)
# webdriver::install_phantomjs()
pjs <- run_phantomjs()
pjs
```

```
## $process
## PROCESS 'phantomjs', running, pid 295222.
##
## $port
## [1] 7143
```

```
ses <- Session$new(port = pjs$port)
```

# Exemplo mínimo

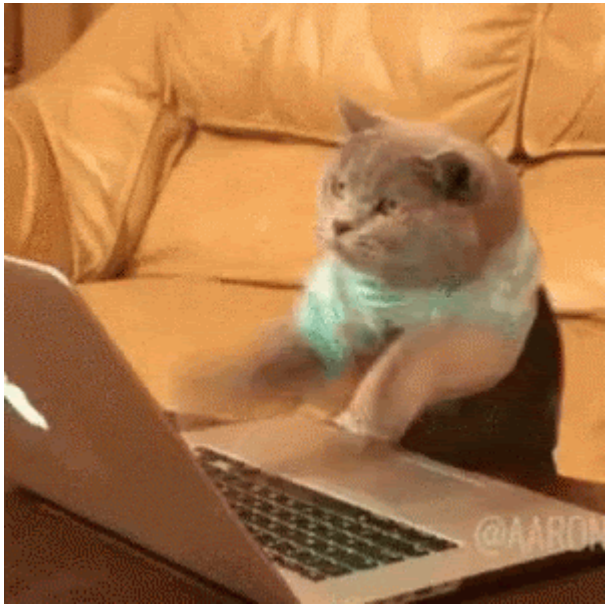
```
ses$go("https://google.com")
ses$takeScreenshot(file = arq)
```



# Elementos

- `ses$findElement()` retorna um elemento da página dado um seletor ou XPath para o mesmo
    - É uma função embutida na sessão (assim como `takeScreenshot()` e `go()`)
- `elem$click()` clica em um elemento, enquanto `elem$sendKeys()` "envia" uma tecla para o elemento
    - São funções embutidas no elemento retornado por `findElement()`
    - A lista `key` contém uma série de teclas que podem ser enviadas (como ENTER, etc.)
    - Ao invés de `elem$sendKeys()` podemos usar `elem$setValue()` para escrever um texto no elemento caso isso seja possível

# Vamos ao R!



# Quer mais?

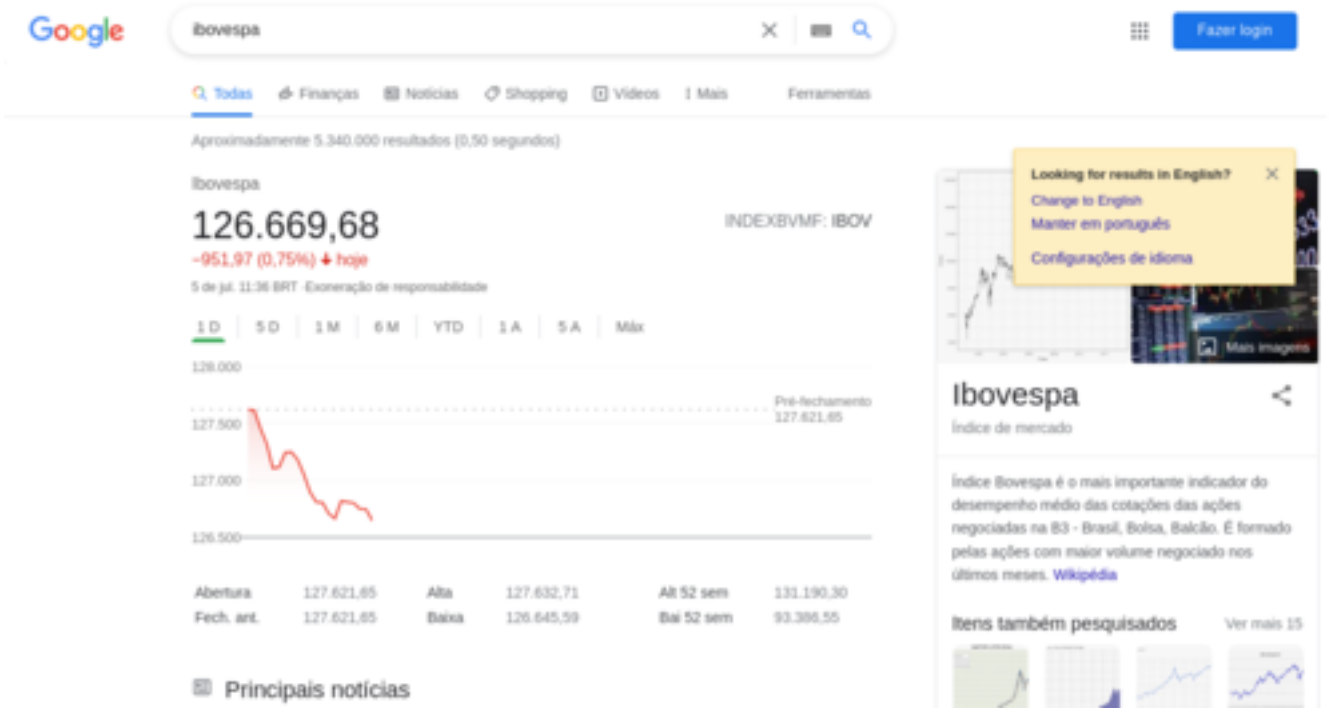
- O PhantomJS, apesar de muito capaz, não consegue exibir todo o conteúdo dinâmico de uma página
- Para solucionar esse problema, é necessário usar o RSelenium com um navegador de verdade como backend
    - Nem sempre a instalação do `RSelenium` funciona e em alguns sistemas operacionais há outras dependências
    - A documentação do `RSelenium` é bagunçada, dificultando qualquer pesquisa
    - O método sugerido para utilizar navegadores externos depende do Docker, um programa sem relação com o R
- Não use `RSelenium` caso não seja estritamente necessário!

# Demonstração

- As funções do `RSelenium` são parecidas com as do `webdriver`, mas envolvem mais esforço
- No exemplo abaixo, o `RSelenium` abre uma aba do Firefox no meu computador e executa todos os comandos ao vivo nela

```
library(RSelenium)
drv <- rsDriver(browser = "firefox", verbose = FALSE)
drv$client$navigate("https://google.com")
elem <- drv$client$findElement("xpath", "//input[@name='q']")
elem$sendKeysToElement(list("ibovespa", key = "enter"))
Sys.sleep(2)
drv$client$screenshot(file = arq)
```

# Demonstração

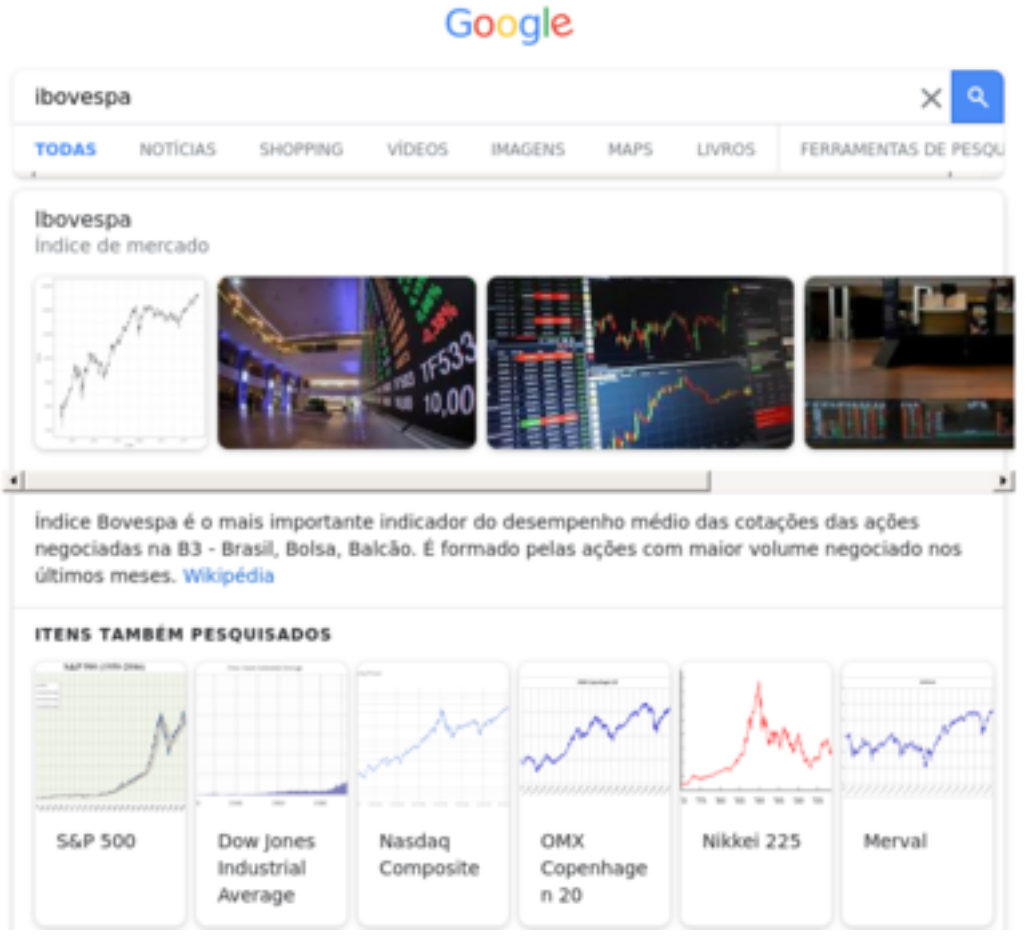


# Mas com o webdriver...

- Note a presença do gráfico interativo na imagem anterior, isso não é possível com o `webdriver`
- Pelas limitações do PhantomJS, nem todo elemento dinâmico pode ser renderizado na tela
    - É possível usar o `webdriver` com Docker também, mas nesse caso é melhor recorrer ao `RSelenium`

```
ses$go("https://google.com")
elem <- ses$findElement(xpath = "//input[@name='q']")
elem$sendKeys("ibovespa", key$enter)
Sys.sleep(2)
ses$takeScreenshot(file = arq)
```

# Mas com o webdriver...




curso-r

# Vamos ao R!