Clase: Scraping dinámico
FCE - UBA
Ciencia de Datos para Economía y Negocios
Nicolás Sidicaro

Clase pasada

Sitio sencillo

- Estático
- Bien ordenado
- Bien nombrado

Cachorros Border Collie Negros Con Blanco

$ 389.999

Mismo precio en 6 cuotas de $ 65.000

Envío gratis

4 colores

Termo Waterdog Ombu 1000ml Negro Ombu1000bk

Por Waterdog

4.7 ★★★★½ (2432)

~~$ 53.041~~

$ 41.902 21% OFF

en 3 cuotas de $ 17.186

Llega gratis mañana

Enviado por FULL

Otra opción de compra

~~$ 60.776~~

$ 50.444 17% OFF

Mismo precio en 6 cuotas de $ 8.407

PHILCO

Smart Tv Philco Pld43fs24vh 43 Pulgadas Led Full Hd

Por Philco

~~$ 499.999~~

$ 334.990 33% OFF

en 3 cuotas de $ 137.401

4.7 ★★★★½ (270)

Envío gratis

Pero

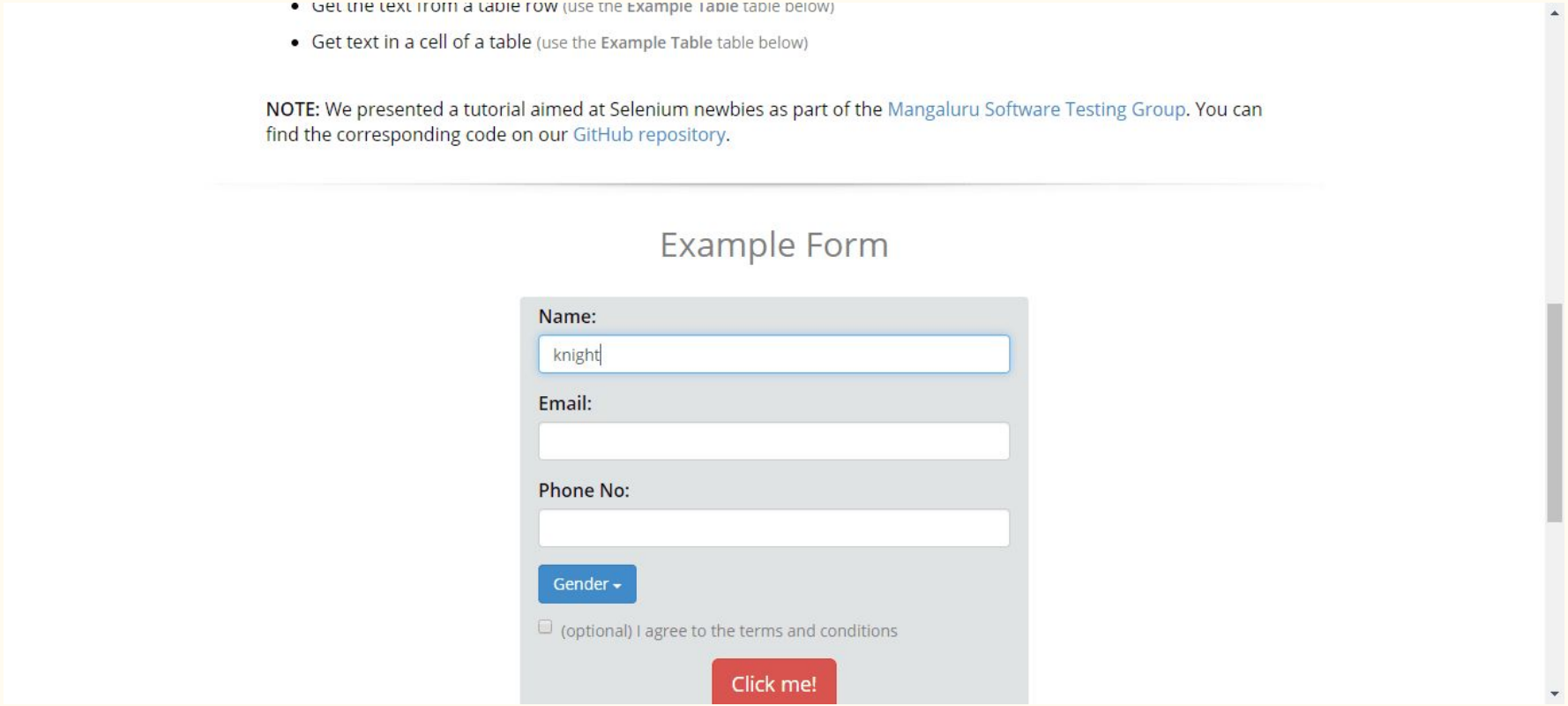
Pero

Pero...

Y si...?

- Hay que llenar un formulario
- Hay que apretar un botón
- La página no se carga entera al ingresar
- Hay scroll infinito
- Si no se ve el contenido, no se carga

Hay que usar Selenium



Selenium

- **¿Qué es Selenium?** Es un framework de código abierto diseñado para automatizar navegadores web a través de diferentes plataformas.
- **Propósito original**: Fue creado inicialmente para realizar pruebas automatizadas de aplicaciones web, pero ha expandido su uso a otras áreas.
- **Diferencia con scraping estático**: A diferencia de métodos como rvest que solo pueden capturar el HTML inicial, Selenium puede interactuar con páginas que cargan contenido dinámicamente mediante JavaScript.

Selenium

Funcionamiento básico: Controla un navegador real, permitiendo emular acciones humanas como:

- Hacer clic en elementos
- Completar formularios
- Desplazarse por la página
- Esperar a que elementos se carguen
- Manejar eventos JavaScript

Casos de uso para scraping dinámico

- Contenido que aparece tras interacciones (clics, scroll)
- Aplicaciones de una sola página (SPA)
- Sitios que requieren inicio de sesión
- Elementos que aparecen después de tiempos de espera

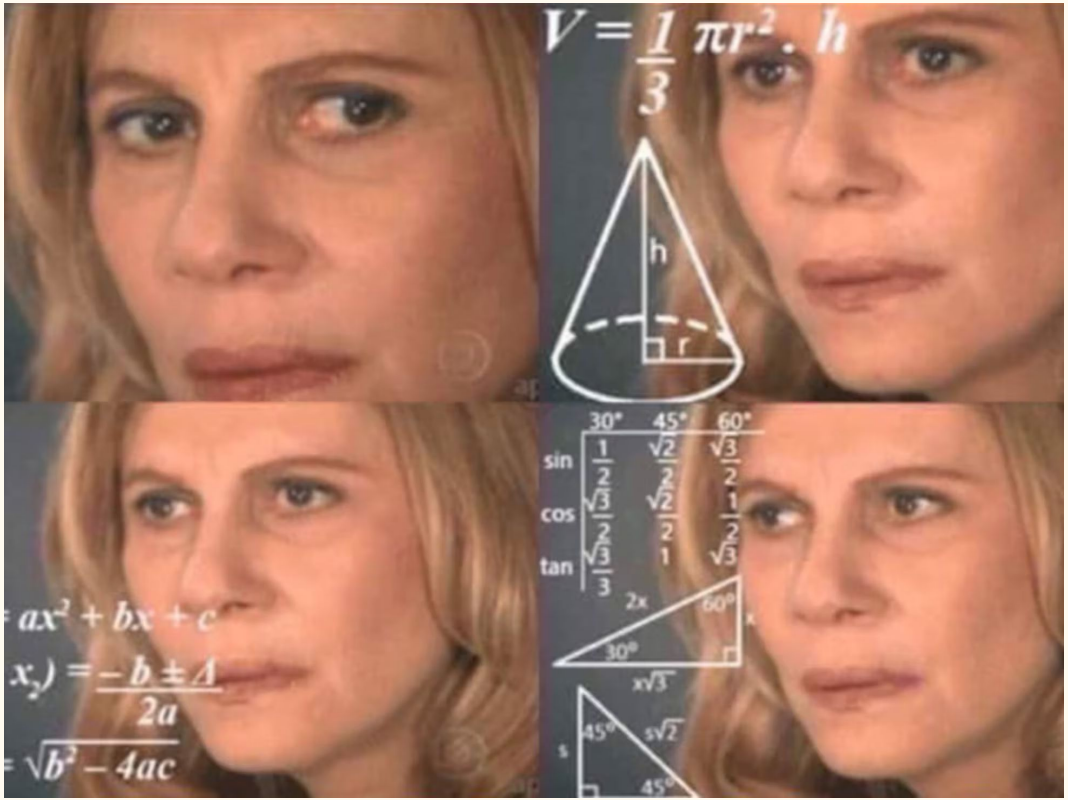
O también... URLs complicadas

Fácil:
https://listado.mercadolibre.com.ar/cafetera#D[A:cafetera]

No tan fácil:
https://listado.mercadolibre.com.ar/electrodomesticos-aires-ac/pequenos-electrodomesticos/cocina/cafeteras/cafetera_Desde_51_NoIndex_True

¿Qué pasa si cambiamos de producto? Cambia mucho la URL
https://listado.mercadolibre.com.ar/camaras-accesorios/camaras/camaras-digitales/camara-de-fotos-reflex_Desde_49_NoIndex_True?sb=all_mercadolibre

O también... URLs complicadas



Entonces

Ventajas principales:

- Acceso a contenido que solo existe después de la ejecución de JavaScript
- Capacidad para automatizar secuencias complejas de interacciones
- Posibilidad de capturar capturas de pantalla durante el proceso

Desafíos:

- Mayor complejidad técnica que el scraping estático
- Ejecución más lenta que métodos como rvest
- Requiere más recursos del sistema

Selenium, RSelenium

Selenium, RSelenium

RSelenium es un paquete de R que proporciona una interfaz para Selenium WebDriver, permitiendo a los usuarios de R:

- Controlar navegadores web programáticamente desde el entorno R
- Automatizar interacciones con páginas web dinámicas que dependen de JavaScript
- Realizar scraping de contenido dinámico que no es accesible con métodos de scraping estático tradicionales
- Emular comportamiento humano en sitios web (clics, completar formularios, scroll, etc.)
- Extraer información de páginas protegidas por logins o que requieren interacción
- Conectar el ecosistema R con el poder de Selenium WebDriver, ampliamente utilizado en testing y automatización web

Es esencialmente un "puente" que conecta las capacidades de análisis de datos de R con las potentes herramientas de automatización de navegadores de Selenium.

Selenium, RSelenium

- Lo vamos a correr sobre Firefox, suele ser el más compatible
- Quizás tengan que instalar Java o algo para que funcione, me pueden ir consultando o hablar con la IA para que les sugiera pasos si aparece un error

Bastante más que rvest: navegación

- **rsDriver()**: Función clave para iniciar un servidor Selenium y cliente en modo remoto
- **remoteDriver()**: Alternativa para conectarse a un servidor Selenium ya en ejecución
- **navigate()**: Para dirigirse a una URL específica
- **getCurrentUrl()**: Obtiene la URL actual
- **goBack(), goForward()**: Para navegar en el historial del navegador
- **refresh()**: Actualiza la página actual

Bastante más que rvest: localizar elementos

- **findElement():** encuentra el primer elemento que coincide con un selector

  elemento <- remDr$findElement(using = "css", value = ".clase")

- **findElements():** Encuentra todos los elementos que coinciden con un selector (devuelve lista)

  elementos <- remDr$findElements(using = "xpath", value = "//div[@class='producto']")

Bastante más que rvest: localizar elementos

Métodos de selección (using parameter):

- "css selector" - Para selectores CSS - también funciona CSS
- "xpath" - Para expresiones XPath
- "id" - Por atributo ID
- "name" - Por atributo nombre
- "class name" - Por clase
- "tag name" - Por etiqueta HTML
- "link text" - Por texto de enlace exacto
- "partial link text" - Por texto parcial de enlace

Info: https://cran.r-project.org/web/packages/RSelenium/RSelenium.pdf#page=8

Bastante más que rvest: interactuar con elementos

- **clickElement()**: Hace clic en un elemento
    - elemento$clickElement()

- **sendKeysToElement()**: Envía texto a campos de entrada
    - inputElement$sendKeysToElement(list("texto a introducir"))

- **clearElement()**: Borra el contenido de un campo de texto

- **submitElement()**: Envía un formulario

- **isElementDisplayed()**: Verifica si un elemento es visible

- **isElementEnabled()**: Verifica si un elemento está habilitado

- **isElementSelected()**: Verifica si un elemento está seleccionado (para checkboxes/radios)

Bastante más que rvest: obtener información

- **getPageSource():** Obtiene el código HTML completo de la página actual

```
html <- remDr$getPageSource()[[1]]
# Se puede pasar a rvest para análisis
pagina <- read_html(html)
```

- **getElementText():** Obtiene el texto de un elemento

```
texto <- elemento$getElementText()[[1]]
```

- **getElementAttribute():** Obtiene un atributo específico de un elemento

```
href <- elemento$getElementAttribute("href")[[1]]
```

Bastante más que rvest: cerrar

- **close():** Cierra la ventana actual del navegador
- **closeServer():** Cierra el servidor Selenium (con rsDriver)
- Cerrar sesión de R

Veamos más de cerca el código

- **Caso:** MercadoLibre
- **Diferencias con clase anterior**: vamos a utilizar el buscador y recorrer todas las páginas de cada producto buscado
- **¿Podría hacerse con rvest?** Sí, pero a veces es más fácil así
- Los pasos a seguir son parecidos a los de la clase pasada: identificar elementos, seleccionarlos, levantar el texto

Ejercicio en clase

- **Caso:** https://www.fravega.com/ (Ahora sí!)

- Productos a buscar:
  - Búsqueda: misma que está en el código visto en clase hoy

- ¡Ojo!
  - El selector del cuadrado de búsqueda es el que está dentro del cuadrado (ya van a entender)
  - Borren la parte de “Tiendas Oficiales” del código de MercadoLibre, ya que no aplica a este caso