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

Web Scraping con R: Trabajando con rvest

- rvest: Paquete en R diseñado para web scraping, inspirado en la biblioteca Beautiful Soup de Python
- Desarrollador: Hadley Wickham (también creador de tidyverse)
- Funcionalidad: Extracción de datos estructurados de páginas web HTML
- Integración: Funciona perfectamente con el ecosistema tidyverse

Web Scraping con R: Trabajando con rvest

```
# Instalación
install.packages("rvest")

# Carga del paquete
library(rvest)
```

Funciones fundamentales de rvest

Lectura y navegación

- `read_html(url)`: Descarga y parsea una página web
- `html_element(s)`: Selecciona elementos específicos del HTML
- `html_text2()`: Extrae el texto de elementos HTML (recomendado sobre html_text)
- `html_attr(s)`: Extrae atributos de elementos HTML (href, src, etc.)
- `html_table()`: Extrae tablas HTML directamente a data frames


AÚN HAY MÁS!!!

Hay más, pero no nos interesan ahora

Recursos para explorar

- `Documentación rvest`: aquí
- `Intro breve a rvest`: aquí
- `Mayor detalle`: aquí
- `Paquete polite`: para cumplir las normas*. Aquí.

`*No les voy a mentir, no lo uso`


LOONEY TUNES
"That's all Folks!"

Mentira, veamos algún sitio

Mismo sitio:

- distintos formatos
- distintos datos descargables

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

Envío gratis

4.7 ★★★★½ (270)

Caso 1: televisión ¿qué tenemos?

Caso 2: perritos ¿qué tenemos?

¡Ojo! Si la página puede vender en dólares también se podría levantar la moneda de venta (por ejemplo, MercadoLibre en Uruguay vende en las dos monedas y las publicaciones pueden estar en cualquiera de los dos)

¿La imagen se puede levantar? ¿Qué opinan?

Caso 3: termo

Digan ustedes...

-
-

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

¡Formatos distintos implican códigos distintos!

- Si los productos están en horizontal hay un tipo de código
- Si están en vertical, otro tipo
- ¿Por qué?

¡Formatos distintos implican códigos distintos!

- Si los productos están en horizontal voy a hacer un código de R
- Si están en vertical, otro código
- ¿Por qué? Son distintos HTMLs y distintos CSS
- En realidad, es un mismo código con las dos opciones

¡Formatos distintos implican códigos distintos!

- Si los productos están en horizontal voy a hacer un código de R
- Si están en vertical, otro código
- ¿Por qué? Son distintos HTMLs y distintos CSS
- En realidad, es un mismo código con las dos opciones
- Vamos a necesitar combinar condiciones (*if*) con el *scraper* para que todo funcione bien

¿Qué sucede si quiero *scrapear* muchas páginas del mismo sitio?

- ¿Tienen la misma estructura?
- ¿Ya sabes cómo cambia la URL o tenés el listado de URLs?
- Si las dos respuestas son sí ¿qué necesitamos hacer?

¿Qué sucede si quiero *scrapear* muchas páginas del mismo sitio?

- ¿Tienen la misma estructura?
- ¿Ya sabes cómo cambia la URL o tenés el listado de URLs?
- Si las dos respuestas son sí ¿qué necesitamos hacer?
  - For Loops: iterar una y otra vez sobre las páginas que aparezcan. Si tiene el mismo formato, funciona
  - ¿Qué pasa si se rompe porque cambia algo de la página? Se rompe el loop
  - A menos que usemos Try Catch

¿Qué es Try-Catch?

- Estructura de control de errores que "atrapa" excepciones para evitar que el programa se detenga
- En R se implementa con las funciones tryCatch() o try()
- Permite manejar errores de forma elegante y continuar la ejecución
- Esencial para operaciones propensas a fallar, como peticiones web

¿Por qué usarlo en *web scraping*?

- **Alta probabilidad de errores:** Conexiones interrumpidas, cambios en la estructura de la página, timeouts
- **Protección del bucle:** Evita que todo el proceso se detenga por un solo error
- **Datos parciales:** Permite obtener algunos datos incluso si parte del scraping falla
- **Documentación de problemas:** Registra qué URLs o elementos específicos causaron problemas

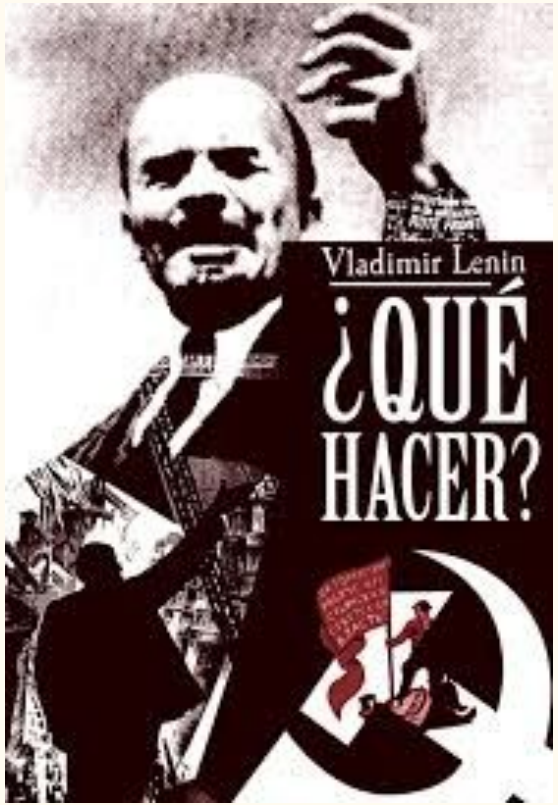
¿Por qué usarlo en *web scraping*?

```r
# Vector para almacenar resultados
resultados <- vector("list", length(urls))
errores <- character(length(urls))

# Bucle con Try-Catch
for(i in seq_along(urls)) {
  # Pausa para no sobrecargar el servidor
  Sys.sleep(2)

  # Try-Catch para manejar errores
  resultados[[i]] <- tryCatch({
    # Código que podría fallar
    pagina <- read_html(urls[i])
    datos <- pagina %>%
      html_elements(".producto") %>%
      html_text2()

    # Si llega aquí, no hubo errores
    errores[i] <- NA
    datos  # Esto será asignado a resultados[[i]]

  }, error = function(e) {
    # Manejo del error
    errores[i] <<- paste("Error en URL", i, ":", e$message)
    message(errores[i])  # Mostrar mensaje de error
    return(NULL)  # Valor para resultados[[i]] cuando hay error
  })
}
```

- Envuelve todo el código susceptible de errores
- Permite correr algo y que el error se muestre y/o guarde
- Una vez que ocurra, el loop va a seguir con la próxima iteración
- Nos salva del proceso: correr código - irse a dormir - despertarse sin resultados

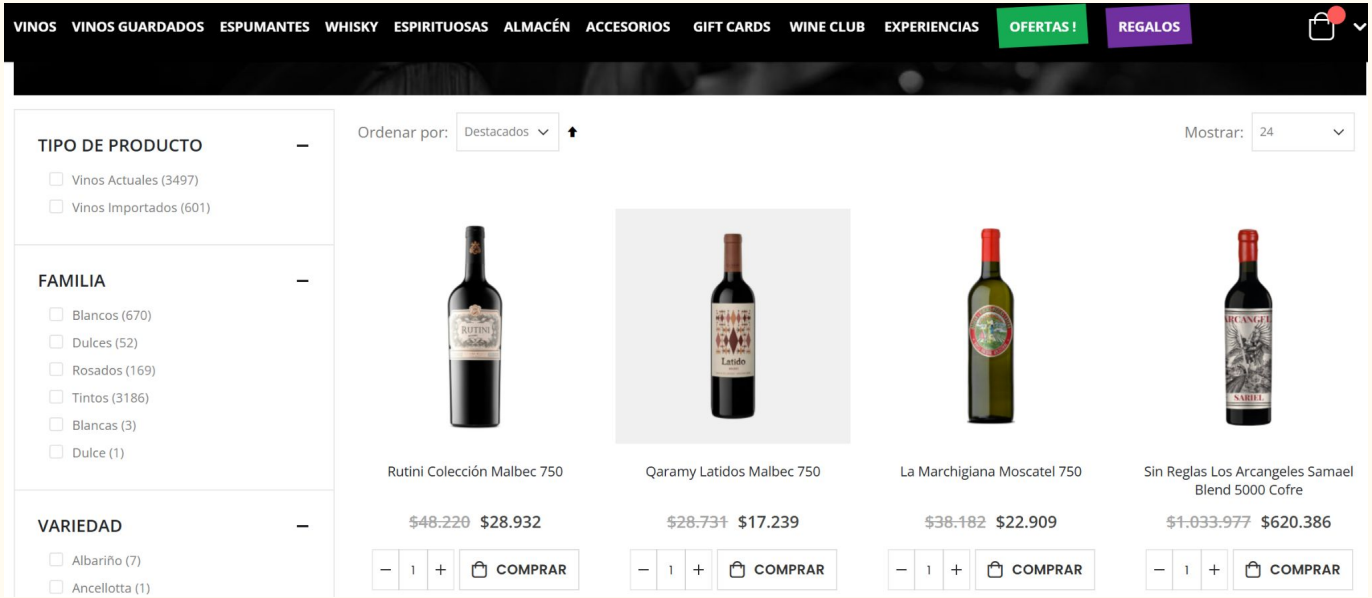
¿Por qué usarlo en *web scraping*?

¿Qué hacer en *Web Scraping*?



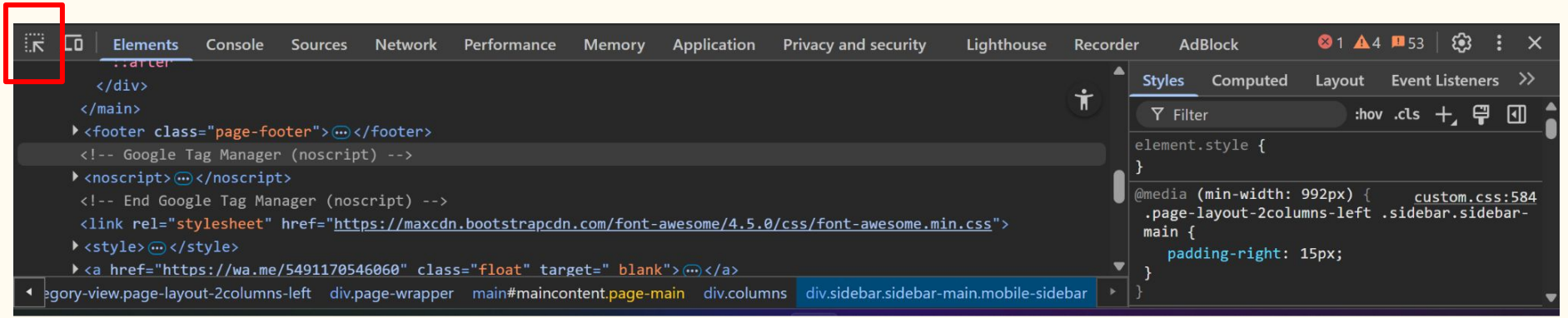
- ¿Cómo es la URL? ¿Cómo cambia?
- ¿Es estática o dinámica?
- ¿Tiene algún freno de scrapers? Ej: zonaprop tira error. Crunchbase tiene Cloudfire
- ¿Tiene consistencia el sitio?
- Detectar elementos principales y qué datos queremos levantar
- Ver detalle en clase “Scraping_clase_1” sección “Checklist para análisis de sitio web”

Piano piano... Vinoteca Ligier



- Si hacemos scroll hasta abajo, vemos que hay páginas
- Si cambiamos de página vemos que se agrega el parámetro de página: https://vinotecaligier.com/vino?p=2
- En el costado izquierdo dice la cantidad total de vinos

Piano piano... Vinoteca Ligier



- Apretamos botón derecho en alguna parte de la página y presionamos inspeccionar
- Luego, apretamos el botoncito del lado izquierdo para ver el html
- Y nos posamos sobre alguno de los boxes de vinos para identificar el nombre del contenedor clave: el que tiene adentro todos los datos de ese vino
- Chequeamos que se repita la cantidad de vinos que haya publicados

Piano piano... Vinoteca Ligier

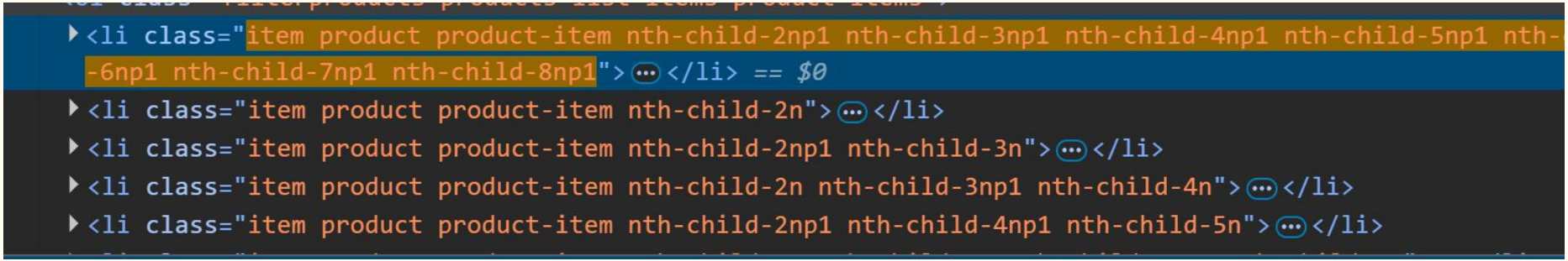
Trivento Established Semillon 750

Humberto Canale Blush Rose Dulce 750

Marlborough Sun Riesling 750

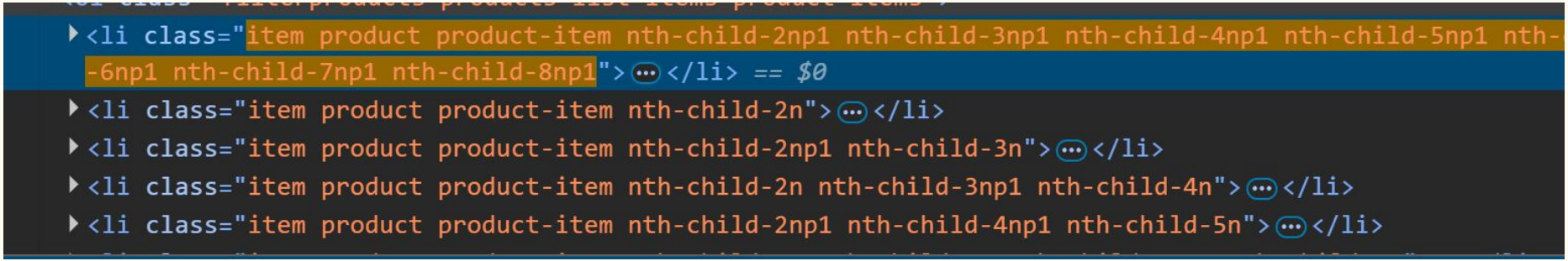
Marlborough Sun Sauvignon Blanc 750

Piano piano...



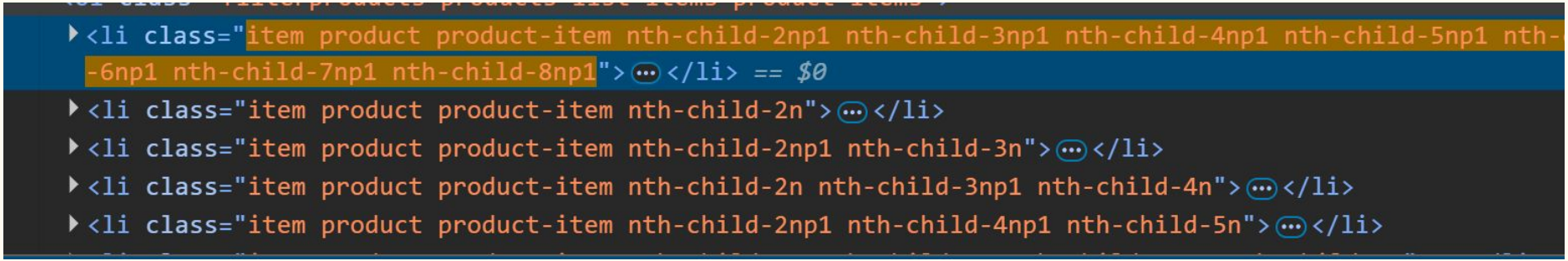
- El que hizo esta página es el representante del Maligno en la Tierra
- Todas las cajitas se llaman distinto
- Pero tienen algo en común...

Piano piano...



- El que hizo esta página es el representante del Maligno en la Tierra
- Todas las cajitas se llaman distinto
- Pero tienen algo en común... todas empiezan con “item product product”
- Por lo general, utilizo el selector css para scraping, pero como varía tengo que usar xpath

Piano piano...




```
pagina <- read_html('https://vinotecaligier.com/vino')

cuadros <- pagina %>%
  html_elements(xpath = '//li[starts-with(@class, "item product product")]')
```

Una vez que tenemos eso

2 opciones

- Si no nos interesa saber más que el nombre y el precio, podemos levantar la información de ese recuadro
- Si queremos saber las características, tenemos que levantar la URL para poder scrapear la página de detalle
- En ambos casos, la estructura es similar

2 opciones

Opción 1: sin detalle

```
for(i in 1:length(cuadros)){
  tryCatch({
  nombre <- cuadros[[i]] %>%
    html_elements(css='[class="product-item-link"]') %>%
    html_text2()
  precio_viejo <- cuadros[[i]] %>%
    html_elements(css='[class="old-price"]') %>%
    html_text2()
  precio_nuevo <- cuadros[[i]] %>%
    html_elements(css='[class="special-price"]') %>%
    html_text2()
  todo <- cuadros[[i]] %>%
    html_text2()
```

2 opciones

Opción 2: con detalle - necesito obtener la URL del vino

```
for(i in 1:length(cuadros)){
  tryCatch({
  url <- cuadros[[i]] %>%
    html_elements(css='[class="product-item-link"]') %>%
    html_attr('href')
```

Ejercicio en clase

1) Siguiendo el código visto en clase, replicar el scraping para la primera página de https://www.espaciovino.com.ar/vinos?t=Tinto&v=Malbec&o=recomendados

2) ¿Cómo harías para levantar todos los precios de las páginas?

3) Ingresá a Cuit Online, poné el siguiente CUIT: 30701307115
   a) ¿Qué información útil podés sacar?
   b) ¿Cómo harías para buscar otro CUIT?
   c) Se puede generalizar el procedimiento?