Datos Abiertos (*Open Data*)

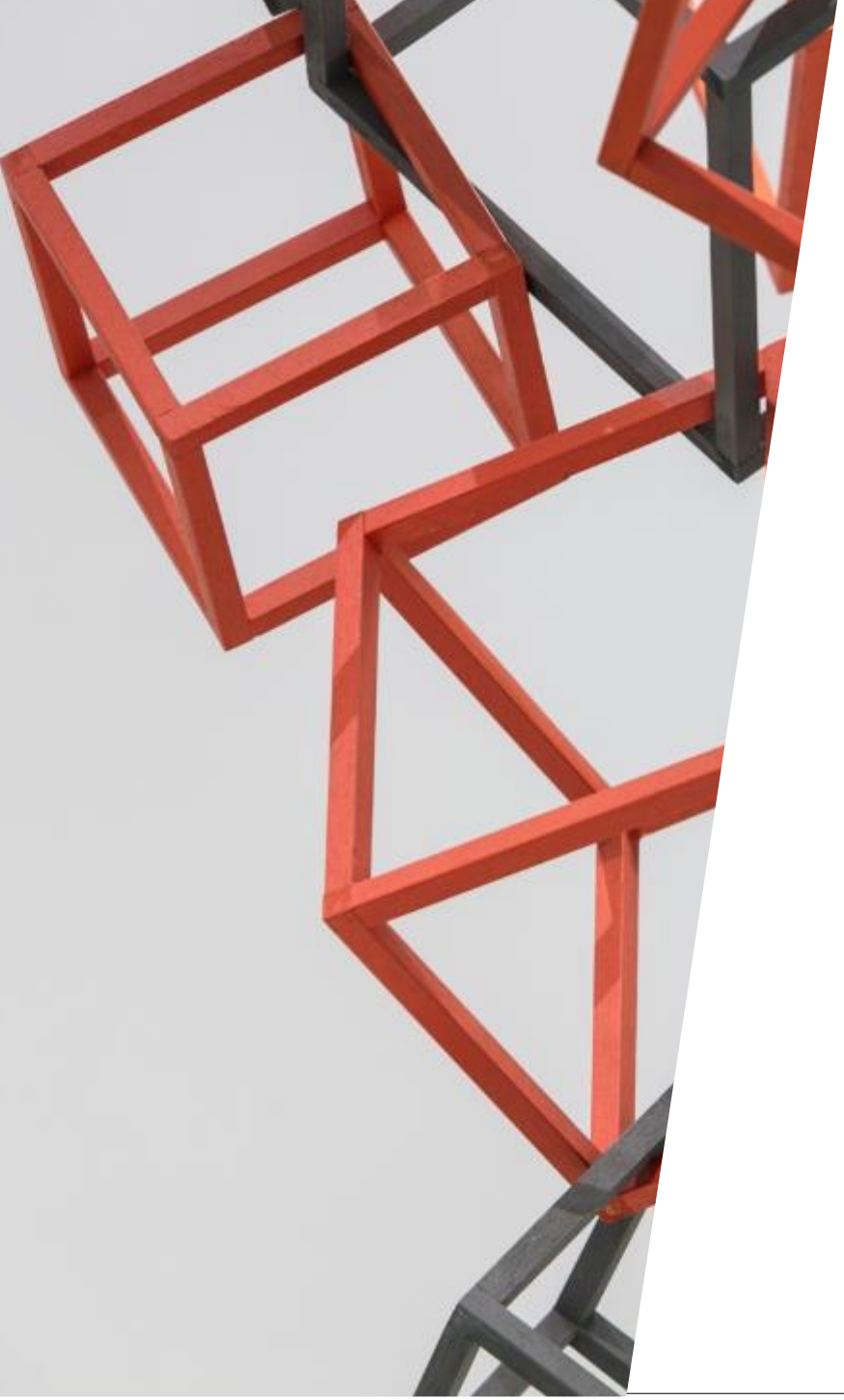
Máster en Data Science y Big Data en Finanzas (MDS_F)
Máster en Data Science y Big Data (MDS)

José Manuel Rodríguez Madrid
jmrodriguez@afi.es

Afi Escuela

Índice

¿Qué son los Datos Abiertos y por qué son valiosos?



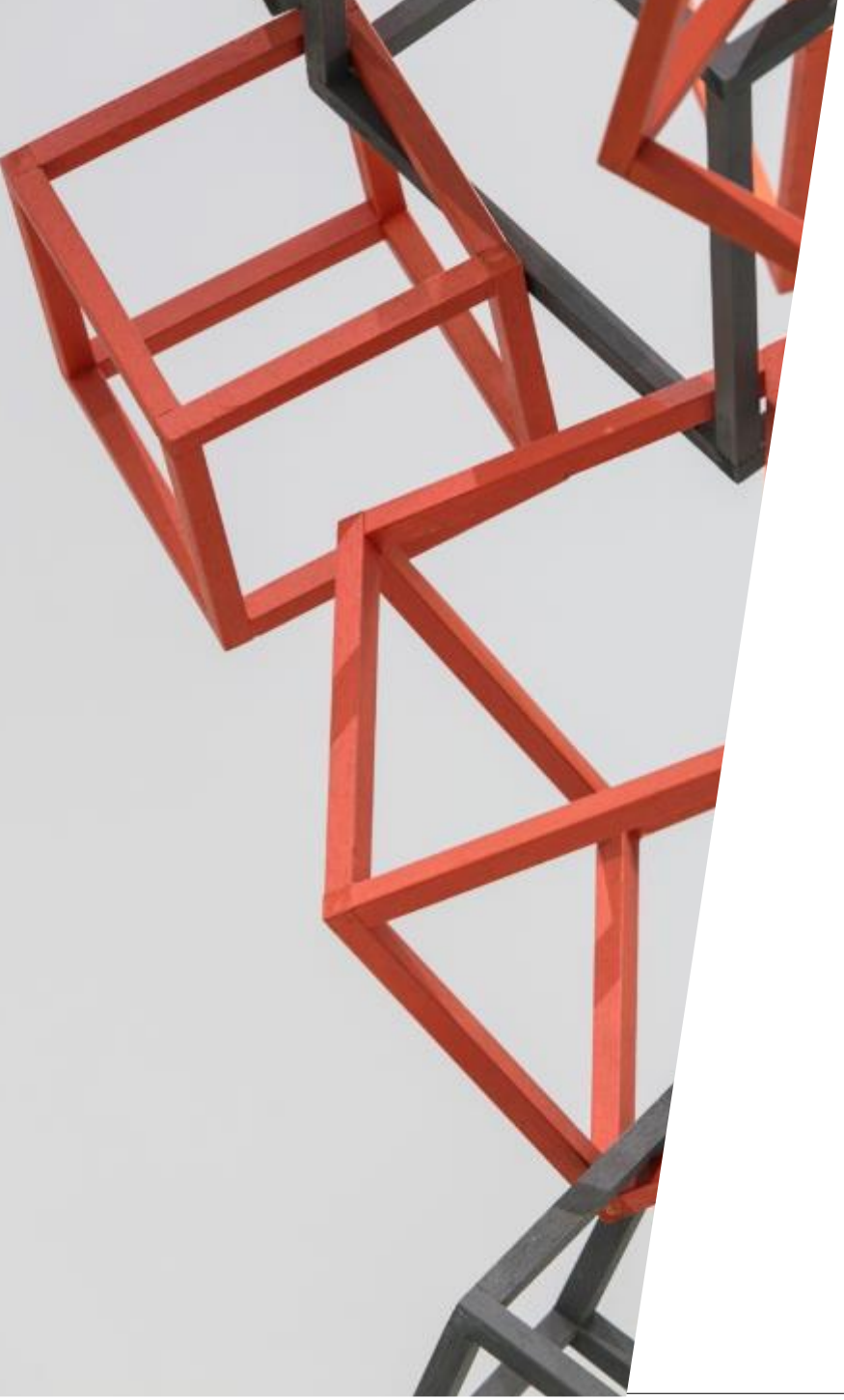
¿Qué son los Datos Abiertos?

- *"Open data is data that can be **freely used, re-used and redistributed by anyone** - subject only, at most, to the requirement to attribute and sharealike."* (1)

- *"Los datos abiertos son datos que pueden ser **utilizados, reutilizados y redistribuidos libremente por cualquier persona**, y que se encuentran sujetos, cuando más, al requerimiento de atribución y de compartirse de la misma manera en que aparecen."* (2)

(1) Open data handbook. (s.f.). *What is Open Data?* http://opendatahandbook.org/guide/en/what-is-open-data/

(2) Open data handbook. (s.f.). ¿Qué son los Datos Abiertos? http://opendatahandbook.org/guide/es/what-is-open-data/



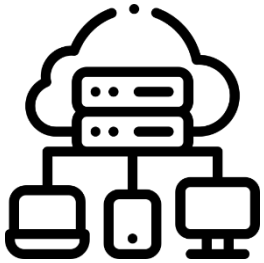
¿Qué son los Datos Abiertos?

"Open data and content can be ***freely used, modified, and shared*** *by* ***anyone*** *for* ***any purpose.***" (3)

(3) Open definition. (s.f.). *The Open Definition* https://opendefinition.org/

¿Qué son los Datos Abiertos?

Aspectos clave de los Datos Abiertos

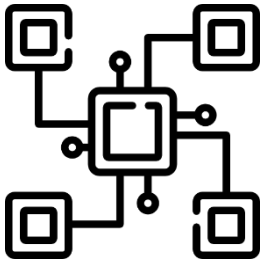


Disponibilidad y accesibilidad

Datos disponible como un todo y a un coste razonable.

Datos disponibles a través de internet.

Datos en formato adecuado.



Reutilización y distribución

Se debe permitir la reutilización y redistribución del conjunto de datos.

Facilidad de integración con otras fuentes de datos.

Universalidad

Los datos pueden ser utilizados, reutilizados y distribuidos por todos, sin restricciones de ningún tipo.

¿Por qué son valiosos los Datos Abiertos?

Interoperabilidad

- "*Habilidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada*".

- Unos Datos Abiertos de calidad garantizan poder combinar y mezclar distintos conjuntos de datos.

- Gracias a esto podemos **construir sistemas más complejos y grandes**, permitiendo desarrollar nuevos productos y servicios.

- Unos Datos Abiertos que no facilitan la interoperabilidad no podrán obtener el benefició de integrarse en estos sistemas más complejos y grandes, que es donde reside el valor.

Ejemplo: ¿dos conjuntos de datos que manejan la entidad *municipio* pero que los identifican con códigos diferentes podrán interoperar?

Ejemplos de Datos Abiertos

Open Science

A favor

- Ejercicio de transparencia. Dado que la mayoría de los estudios de investigación son financiados con dinero público, el resultado final de dichas investigaciones debe ser accesible para la sociedad.
- Aumenta la rigurosidad y reduce los errores de las investigaciones, al ser abiertas al público para su escrutinio y revisión.
- Hace que los trabajos de investigación sean más reproducibles.

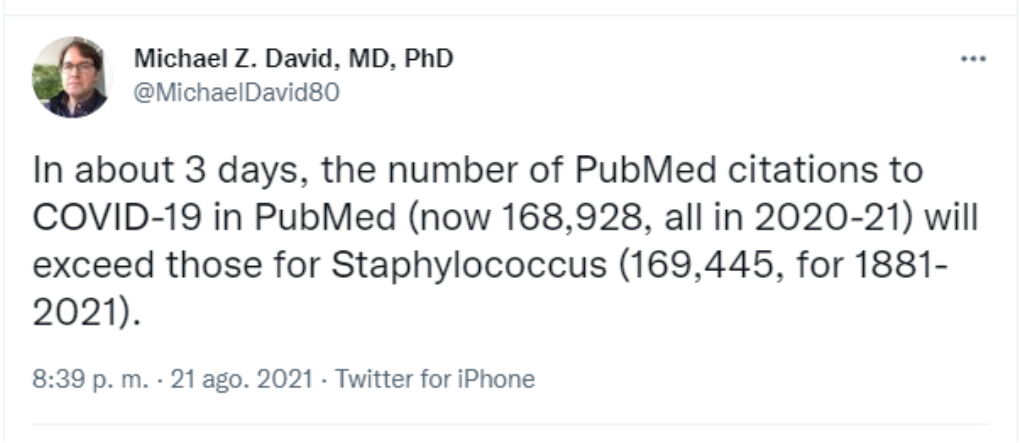
En contra

- Preferencia de ciertos investigadores por no conocer otros estudios en su campo y así encontrar motivación para sus propias investigaciones.
- Determinados estudios científicos pueden emplearse con fines perversos.
- Los estudios y datos derivados de estudios científicos pueden ser malinterpretados.
- Puede entorpecer la verificación de los estudios, haciendo más lento el proceso.

(4) John P. A. Ioannidis (2005). Why Most Published Research Findings Are False.
https://journals.plos.org/plosmedicine/article?id=10.1371/journal.pmed.0020124

Ejemplos de Datos Abiertos

Open Science



(5) Michael Z. David, MD, Phd. (2021). https://twitter.com/MichaelDavid80/status/1429151259370696710?s=20

(6) Adam Miller, CBC News. (2021). A Canadian COVID-19 study that turned out to be wrong has spread like wildfire among anti-vaxxers. https://www.cbc.ca/news/health/covid-19-vaccine-study-error-anti-vaxxers-1.6188806

Ejemplos de Datos Abiertos

Open Goberment

A favor

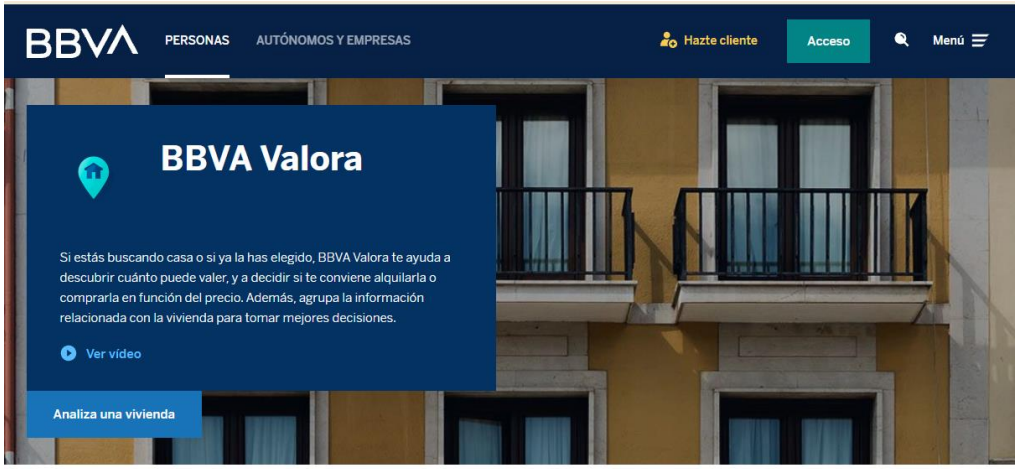
- Transparencia y control democrático.
- Participación ciudadana.
- Mejora de los servicios públicos actuales (eficiencia y eficacia).
- Creación de servicios públicos nuevos.
- Innovación (compañías que emplean los Datos Abiertos para la creación de nuevos productos).
- Medición del impacto de las políticas.
- Extracción de conocimiento a partir fuentes de datos y análisis de patrones sobre grandes volúmenes de datos (*Big Data*).

En contra

- No se debería emplear financiación publica en actividades que compitan con empresas privadas.
- Si hay iniciativas privadas que hacen uso de datos públicos generados por las administraciones públicas éstas deberían pagar por su uso.
- Garantizar la privacidad de datos sensibles de los ciudadanos.
- Seguridad. Algunos datos gubernamentales no podrán ser liberados por motivos de seguridad nacional.

Ejemplos de Datos Abiertos

Open Goberment



(7) BBVA. (s.f.). *BBVA Valora.* https://www.bbva.es/personas/experiencias/bbva-valora/hogar.html

(8) CIVIO. (s.f.). *¿Dónde van mis impuestos?* https://dondevanmisimpuestos.es/politicas#view=functional&year=2021

¿Cómo “abrir” los datos?

Descripción del proceso

- Elige los datos a abrir.
  - Datos de interés y que sea factible publicarlos.
  - Puedes preguntar a la comunidad (foros, redes sociales, etc.) para ver si tienen interés en los mismos. La comunidad será quien los vaya a utilizar y pueden darte *feedback* útil.
- Busca un licencia abierta.
  - Examina los datos para determinar si están sujetos a derechos de propiedad intelectual.
  - Elige el tipo de licencia que se ajusta mejor a tus datos. (9)
- Hazlos disponibles para la comunidad.
  - Disponibles para descargar desde Internet (alojados en servidores HTTP/FTP, mediante *torrents* o APIs)
  - Usa formatos reutilizables (XLS, CSV, TSV, JSON, XML). Evita formatos como PDF.
  - Intenta que los datos se puedan mantener actualizados en tiempo real y evita que dependan de actualizaciones manuales.
- Hazlos visibles y fácilmente “descubribles”.
  - Cuando más usuarios potenciales potencial.
  - Puedes usar portales para publicarlos: DataHub (10), ProrammableWeb (11), Google (12), Kaggle (13).

(9) Open definition. (s.f.). *Conformant Licenses.* https://opendefinition.org/licenses/
(10) DataHub (s.f.). Search for Datasets. https://datahub.io/search
(11) ProgrammableWeb (s.f.). API Directory. https://www.programmableweb.com/category/all/apis
(12) Google (s.f.). Google Dataset Serach. https://datasetsearch.research.google.com/
(13) Kaggle (s.f.). Kaggle Datasets. https://www.kaggle.com/datasets

¿Cómo “abrir” los datos?

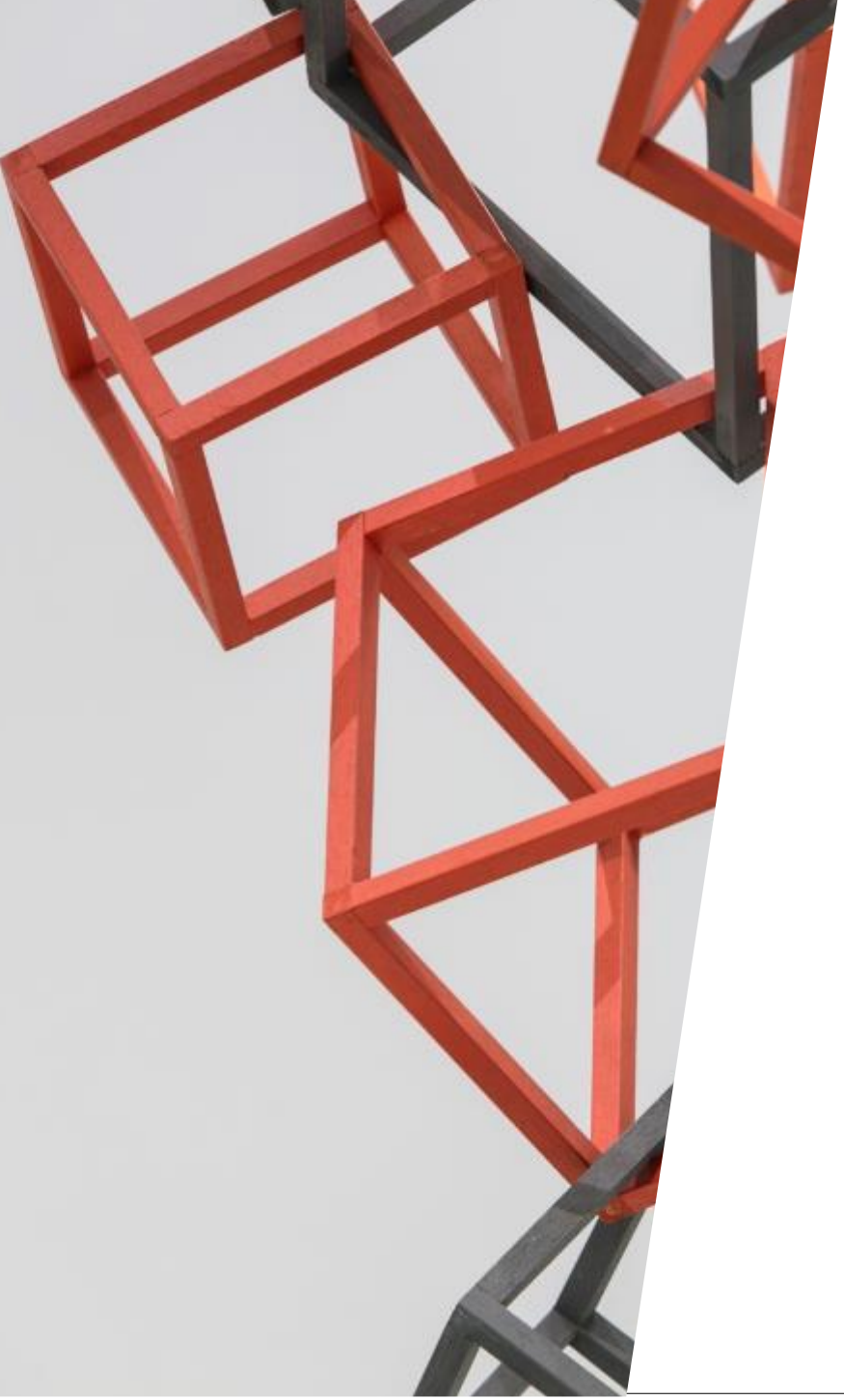
5 ★ Open Data [14]

- Esquema de desarrollo de Datos Abiertos con 5 niveles. Creado por Tim Berners-Lee [15].
- Los niveles son:
  - ★ publica tus datos en Internet (en cualquier formato) y bajo una licencia abierta.
  - ★★ publícalos como datos estructurados (por ejemplo en Excel).
  - ★★★ usa formatos no propietarios (por ejemplo en CSV)
  - ★★★★ emplea URIs para referenciar datos, con estándares W3C (RDF [15] y SPARQL [16]).
  - ★★★★★ enlaza tus datos con otros datos para proveer contexto.
- Los tres primeros niveles garantizan:
  - Datos accesibles en Internet bajo licencia abierta.
  - Con formato estructurado.
  - Accesibles por todo el mundo.
- Los dos últimos niveles añaden:
  - Integración con la web.
  - Datos enlazados con otros datos.

(14) 5 ★Open Data. (s.f.). https://5stardata.info/en/
(15) Wikipedia (s.f.). Tim Berners-Lee. https://es.wikipedia.org/wiki/Tim_Berners-Lee
(16) W3C (s.f.). Resource Description Framework (RDF). https://www.w3.org/RDF/
(17) W3C (s.f.). SPARQL Query Language for RDF. https://www.w3.org/2001/sw/wiki/SPARQL

Open Data

Web scraping



¿Qué es Web scraping?

- *"Proceso de extracción de información de una web de manera automática."*
- Este proceso se realizar mediante *software* y evita procesos manuales.
- Típicamente se recurre a estas técnicas si no se puede obtener la información usando otras tecnologías, como por ejemplo una API web.

Web scraping

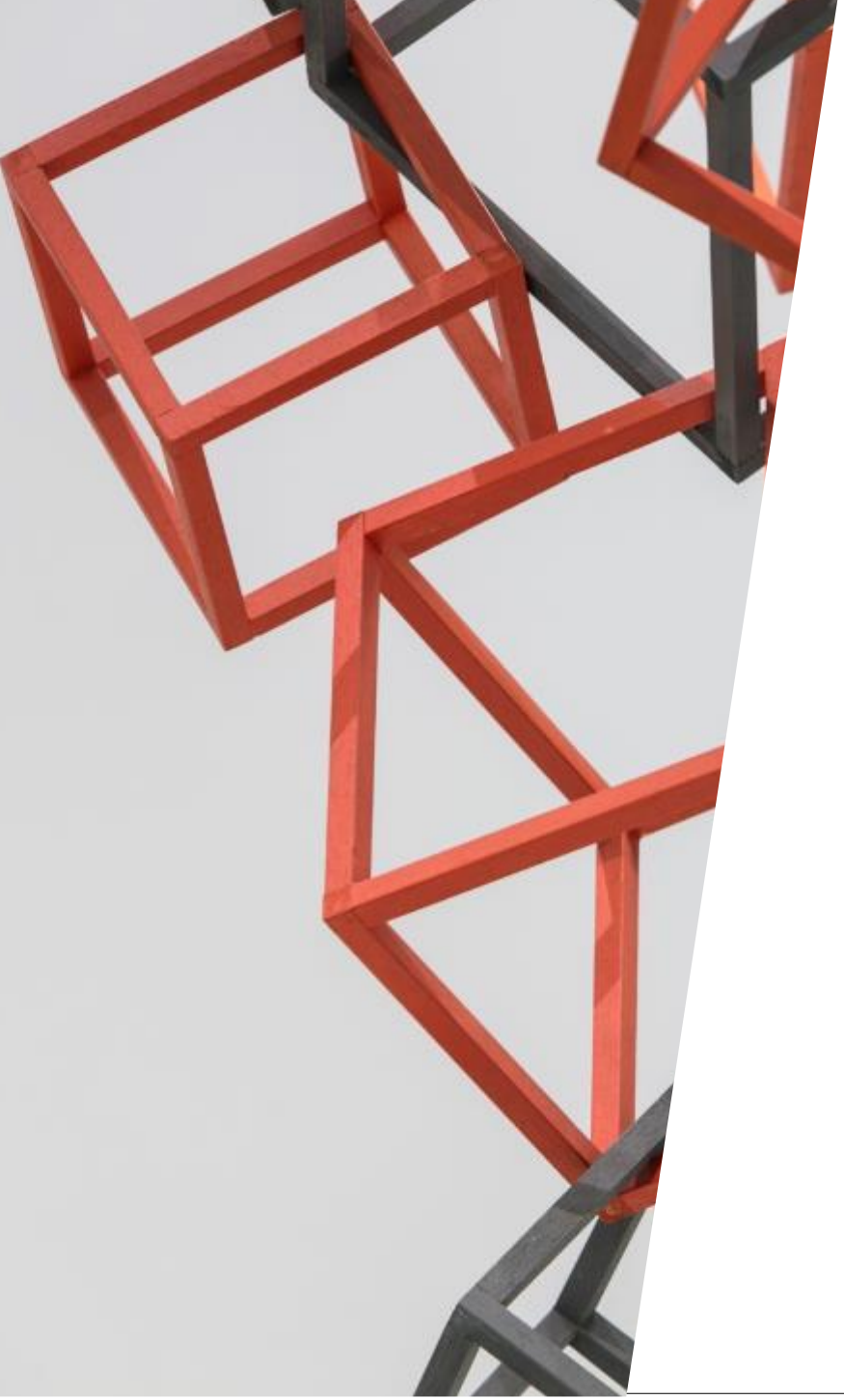
Ventajas e inconvenientes del *web scraping*

Ventajas

- No dependes de un API y sus límites.
- Tienes a tu alcance todos los datos publicados en cualquier web.
- Acelerar el proceso de obtención de datos.

Inconvenientes

- Dependes de la estructura de la página. Si la página cambia el *scraper* dejará de funcionar. Algunas web pueden dificultar el proceso proporcionando la información en formatos como imágenes en lugar de HTML.
- El *web scraping* no siempre es legal. Es necesario comprobar los derechos y propiedad intelectual de los sitios web. Presta atención a los términos legales – ToS (*Terms of Service*) – de la web.
- Es posible que se produzca un bloqueo/*baneo* si se sobrecarga la web a peticiones. Por dos motivos: proteger los datos y proteger los servidores donde se aloja la página web.



Conocimientos necesarios

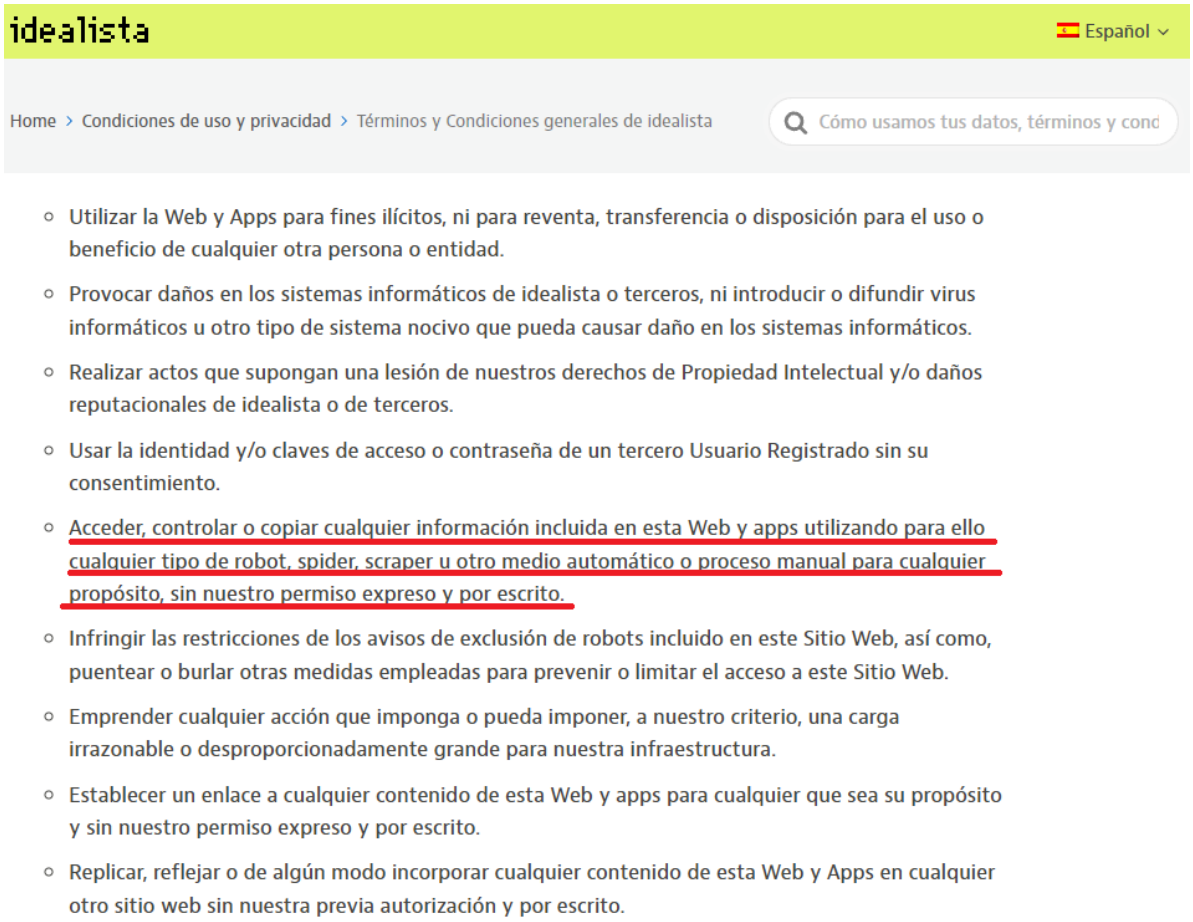
- Conocimientos de programación web.

- Aunque existen herramientas para hacer *web scraping* automático es muy recomendable conocimientos de programación para implementar los *scrapers* en R o Python.

- Conocimientos de expresiones regulares (RegEx). Las expresiones regulares son especialmente útiles cuando se trabaja con información en formato texto.

Web scraping

Términos legales


idealista

Español

Home > Condiciones de uso y privacidad > Términos y Condiciones generales de idealista

Cómo usamos tus datos, términos y cond

- Utilizar la Web y Apps para fines ilícitos, ni para reventa, transferencia o disposición para el uso o beneficio de cualquier otra persona o entidad.
- Provocar daños en los sistemas informáticos de idealista o terceros, ni introducir o difundir virus informáticos u otro tipo de sistema nocivo que pueda causar daño en los sistemas informáticos.
- Realizar actos que supongan una lesión de nuestros derechos de Propiedad Intelectual y/o daños reputacionales de idealista o de terceros.
- Usar la identidad y/o claves de acceso o contraseña de un tercero Usuario Registrado sin su consentimiento.
- Acceder, controlar o copiar cualquier información incluida en esta Web y apps utilizando para ello cualquier tipo de robot, spider, scraper u otro medio automático o proceso manual para cualquier propósito, sin nuestro permiso expreso y por escrito.
- Infringir las restricciones de los avisos de exclusión de robots incluido en este Sitio Web, así como, puentear o burlar otras medidas empleadas para prevenir o limitar el acceso a este Sitio Web.
- Emprender cualquier acción que imponga o pueda imponer, a nuestro criterio, una carga irrazonable o desproporcionadamente grande para nuestra infraestructura.
- Establecer un enlace a cualquier contenido de esta Web y apps para cualquier que sea su propósito y sin nuestro permiso expreso y por escrito.
- Replicar, reflejar o de algún modo incorporar cualquier contenido de esta Web y Apps en cualquier otro sitio web sin nuestra previa autorización y por escrito.

(18) Idealista. (s.f.). *Términos y Condiciones generales de idealista.* https://www.idealista.com/ayuda/articulos/terminos-y-condiciones-generales-de-idealista/

Web scraping

Términos legales

LinkedIn

Condiciones de uso | Política de privacidad | Política de cookies | Política de copyright | Comunicado de privacidad para California

8.2. Lo que no se debe hacer

Aceptas que *no* vas a:

1. Crear una identidad falsa en LinkedIn, falsificar tu identidad, crear un perfil de Miembro para alguien que no seas tú (una persona física), ni usar o intentar usar la cuenta de otra persona.
2. Desarrollar, apoyar o utilizar programas, dispositivos, scripts, robots o cualquier otro medio o proceso (incluidos crawlers, plugins de navegación y complementos, o cualquier otra tecnología) para plagiar los Servicios o copiar de otro modo perfiles u otros datos de los Servicios.
3. Eludir cualquier funcionalidad de seguridad o sortear cualquier control de acceso o los límites de uso del Servicio (como límites en las búsquedas de palabras clave o visualizaciones del perfil).
4. Copiar, utilizar, divulgar o distribuir cualquier información obtenida de los Servicios, ya sea de forma directa o a través de terceros (como los motores de búsqueda), sin el consentimiento de LinkedIn.
5. Revelar información que no tienes derecho a compartir (como información confidencial de terceros, incluida tu empresa).
6. Infringir los derechos de propiedad intelectual de terceros, como derechos de autor, patentes, marcas, secretos comerciales u otros derechos de propiedad. Por ejemplo, no copies ni distribuyas (salvo a través de la funcionalidad de compartir disponible) las publicaciones u otros tipos de contenido de otras personas sin su permiso, el cual se puede otorgar publicando a través de una licencia de Creative Commons.

(19) LinkedIn. (s.f.). *Condiciones de uso.* https://es.linkedin.com/legal/user-agreement

Programación web en 5 minutos

¿Qué es la web?

- **Servidores web:** son máquinas/ordenadores conectados a Internet, los cuales sirven/envían a los clientes documentos/páginas web. Los servidores web pueden ejecutarse también en local (lo cual quiere decir que se ejecutan en la máquina actual en la que estás trabajando).

- **Clientes web:** son programas (también conocidos como navegadores web) que hacen peticiones (*requests*) a los servidores web. Ejemplos de clientes web: Internet Explorer, Edge, Firefox, Chrome, Safari…

- Cada página web puede ser identificada por una dirección URL (*Uniform Resource Locator*) o URI (*Uniform Resource Identifier*). Partes de una URL:

`http://www.afi.es/webAfi/secciones/1131348/Contactanos.html`

  - **Protocolo:** indicador del protocolo de comunicaciones. Por ejemplo: HTTP o HTTPS.
  - **Dominio:** el dominio donde se encuentra almacenado el recurso. Algunas veces pueden aparecer subdominios. Por ejemplo: www, etc.
  - **Información adicional:** como el *path* del recurso solicitado, parámetros, etc.

Programación web en 5 minutos

Funcionamiento de la web

1. Usuario escribe en el navegador la URL solicitada.
2. El navegador resuelve la dirección del dominio y realiza una petición HTTP al servidor web.
3. El servidor web (utilizando los parámetros de la llamadas si los hubiera) localiza/genera el recurso HTML solicitado y lo devuelve al navegador.
4. El navegador tras recibir el HTML, extrae referencias a otros ficheros como hojas de estilo CSS (*Cascading Style Sheets*), imágenes, ficheros JavaScript, etc. y vuelve a pedir cada fichero al servidor web.
5. El servidor web devuelve cada uno de los ficheros anteriores.
6. Finalmente el navegador muestra (*render*) el contenido HTML por pantalla. Este proceso incluye:
    a) Analizar el contenido HTML y extraer el contenido/estructura de la página.
    b) A partir de la hoja de estilos CSS modifica la apariencia (colores, fuentes, márgenes, etc.) a aquellos elementos que sea necesario.
    c) Incluir imágenes en el contenido HTML.
    d) Ejecutar JavaScript, lo que permite alterar el comportamiento de la página HTML.

Programación web en 5 minutos

HTML (*Hypertext Markup Language*)

- Lenguaje de marcas empleado para estructurar el contenido de las páginas web.
- El HTML [20] se almacena en ficheros de texto plano con extensión *.html*.
- La versión actual de HTML es denominada HTML5.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Page Title</title>
    </head>
    <body>
        <h1>Page Title</h1>
        <p>This is a really interesting paragraph.</p>
    </body>
</html>
```

(20) W3Schools. (s.f.). *HTML Tutorial.* https://www.w3schools.com/html/default.asp

Programación web en 5 minutos

HTML: elementos (contenido + estructura)

- Dentro de un documento HTML encontraremos contenido y estructura. A través del lenguaje HTML podremos estructurar el propio contenido.

- HTML se convierte en una herramienta que dota de una *estructura semántica* (jerarquía + relaciones + significado) al contenido.

- HTML no se encarga de la representación visual del contenido del documento. Esta función le corresponde a las hojas de estilo CSS.

- Para dotar de estructura al contenido del documento HTML se añaden elementos [21] definidos por etiquetas o *tags*. Los *tags* de apertura aparecen entre los símbolos **<** y **>**. Los *tags* de cierre aparecen entre **<** y **/>.**

- Las etiquetas se anidan para generar la jerarquía del documento en forma de árbol – **DOM (*Document Objet Model*)**. Es importante asegurar que la apertura y cierre de etiquetas está balanceada para construir un HTML bien formado.

(21) W3Schools. (s.f.). *HTML Elements.* https://www.w3schools.com/html/html_elements.asp

Programación web en 5 minutos

HTML: ejemplo (I)

```
Amazing Visualization Tool Cures All Ills A new open-
source tool designed for visualization of data turns
out to have an unexpected, positive side effect: it
heals any ailments of the viewer. Leading scientists
report that the tool, called D3000, can cure even the
following symptoms: fevers chills general malaise It
is achieves this end with a patented, three-step
process. Load in data. Generate a visual
representation. Activate magic healing function.
```

Programación web en 5 minutos

HTML: ejemplo (II)

```
Amazing Visualization Tool Cures All Ills
```

Título

A new open-source tool designed for visualization of data turns out to have an unexpected, positive side effect: it heals any ailments of the viewer. Leading scientists report that the tool, called D3000, can cure even the following symptoms:

- fevers
- chills
- general malaise

Lista no ordenada

It is achieves this end with a patented, three-step process.

1. Load in data.
2. Generate a visual representation.
3. Activate magic healing function.

Lista ordenada

Programación web en 5 minutos

HTML: ejemplo (III)

```
<h1>Amazing Visualization Tool Cures All Ills</h1>
<p>A new open-source tool designed for visualization of data turns
out to have an unexpected, positive side effect: it heals any
ailments of the viewer. Leading scientists report that the tool,
called D3000, can cure even the following symptoms:</p>
<ul>
    <li>fevers</li>
    <li>chills</li>
    <li>general malaise</li>
</ul>
<p>It is achieves this end with a patented, three-step process.</p>
<ol>
    <li>Load in data.</li>
    <li>Generate a visual representation.</li>
    <li>Activate magic healing function.</li>
</ol>
```

Programación web en 5 minutos

HTML: atributos

- Dentro de los elementos HTML se pueden especificar atributos [22]. Estos atributos son pares clave/valor. Por ejemplo:

```
<tagname property="value"></tagname>
```

- Cada elemento de HTML tiene unos atributos específicos. Por ejemplo: todos los elemento *a* tienen el atributo *href* para indicar la URL a donde enlazan.

- Existen dos atributos especiales que pueden asignarse a cualquier elemento HTML. Estos elementos permiten identificar y manipular elementos del documento HTML.
  - **class:** cada elemento HTML puede tener más de una clase. Además, las clases pueden repetirse entre elementos de la página.
  - **id:** cada elemento HTML puede tener un único id. Además este id debe ser único entre los elementos de la página.

(22) W3Schools. (s.f.). *HTML Attributes.* https://www.w3schools.com/html/html_attributes.asp

Programación web en 5 minutos

CSS: *Cascading Style Sheet*

- Las hojas de estilo o CSS [23] contienen la presentación visual del contenido de un documento HTML. La versión actual de CSS es denominada CSS3.

- Las hojas de estilo están compuestas de selectores y estos a su vez de propiedades.

- Las propiedades son pares clave-valor y se encierran entre llaves.

```
selector {
    property: value;
    property: value;
    property: value;
}
```

- A cada conjunto de selector y propiedades se le conoce como **regla CSS**.

(23) W3Schools. (s.f.). *CSS Tutorial.* https://www.w3schools.com/css/default.asp

Programación web en 5 minutos

CSS: *Selectors* (I)

- Los selectores de CSS permiten identificar los elementos del documento HTML a los que se les aplicarán los estilos. Existen gran cantidad de selectores CSS [24].

- Selectores por tipo (etiquetas HTML)

```
h1     /* todos los elementos de tipo h1 */
p      /* todos los elementos de tipo p */
strong /* todos los elementos de tipo strong */
div    /* todos los elementos de tipo div */
```

- Selectores de descendientes

```
h1 div  /* todos los elementos del tipo div contenidos en un h1 */
div img /* todos los elementos del tipo img contenidos en un div */
```

(24) W3Schools. (s.f.). *CSS Selector Reference.* https://www.w3schools.com/cssref/css_selectors.asp

Programación web en 5 minutos

CSS: *Selectors* (II)

- Selectores por clase (y multiclase)

```
.caption /* todos los elementos con la clase caption */
.label   /* todos los elementos con la clase label */
.axis    /* todos los elementos con la clase axis */

.bar.highlight /* todos los elementos con las clase bar y highlight */
```

- Selectores por ID

```
#header /* el elemento con id "header" */
#title  /* el elemento con id "title" */
```

- Combinación de selectores

```
div.sidebar /* elementos de tipo div con la clase sidebar */
#title.on   /* el elemento con id "title" y la clase on */
```

Programación web en 5 minutos

CSS: properties

- Las propiedades definen la apariencia que tendrá cada elemento del documento HTML. Existen gran cantidad de propiedades CSS [25].

- Ejemplos: margen, relleno, color de fondo, fuente del texto…

```
margin: 10px;
padding: 25px;
background-color: #C00;
font-family: Arial;
```

- Existen tres maneras de incluir el CSS a una página o documento HTML.
  - Embebiendo el CSS dentro del HTML con la etiqueta *style*.
    ```
    <style type="text/css"> p { margin: 10px; background-color: #C00; font-family: Arial; }</style>
    ```
  - A través de una referencia (etiqueta *link*) a un fichero CSS externo (*.css*) dentro del HTML.
    ```
    <link rel="stylesheet" href="style.css">
    ```
  - Incluyendo la regla CSS directamente dentro del elemento HTML. Para ellos se emplea el atributo *style* que tiene todos los elementos HTML. En este caso particular no se hace uso de selectores, puesto que los estilos se aplican directamente al elemento.
    ```
    <p style="margin: 10px; background-color: #C00; font-family: Arial;">Texto de prueba</p>
    ```

(25) W3Schools. (s.f.). *CSS Properties.* https://www.w3schools.com/cssref/default.asp

Programación web en 5 minutos

Javascript

- Lenguaje de *scripting* para hacer páginas web dinámicas. Mediante JavaScript [26]podemos manipular el DOM después de que una página ha sido cargada en el navegador.

- **JavaScript y Java no son lo mismo y no se parecen.**

- El código JavaScript de una página puede estar contenido dentro del propio HTML entre las etiquetas **<script>…</script>**.

```
<body>
        <script type="text/javascript">alert("D3.js is awesome");</script>
</body>
```

- O en un fichero a parte (extensión .js).

```
<script type="text/javascript" src="myscript.js"></script>
```

(26) W3Schools. (s.f.). *Javascript Tutorial.* https://www.w3schools.com/js/default.asp

Paquetes para hacer *web scraping*

Comparativa entre R y Python

R

- Peticiones HTTP (carga HTML): httr
- Procesamiento de HTML: rvest
- Implementación de *crawlers*: Rcrawler
- *Scraping* en web dinámicas: RSelenium

Python

- Peticiones HTTP (carga HTML): requests
- Procesamiento de HTML: BeautifulSoup
- Implementación de *crawlers*: scrapy
- *Scraping* en web dinámicas: selenium

Algunas definiciones técnicas:

- *Spyder:* programa que descarga información de una página web.
- *Crawler*: programa que recorre una web leyendo la información de todas sus páginas. Para ello normalmente se hace *crawling* vertical y horizontal.
- Web dinámica: página web que mediante código Javascript ejecutado en el navegador determina como se cargan sus partes/componentes.

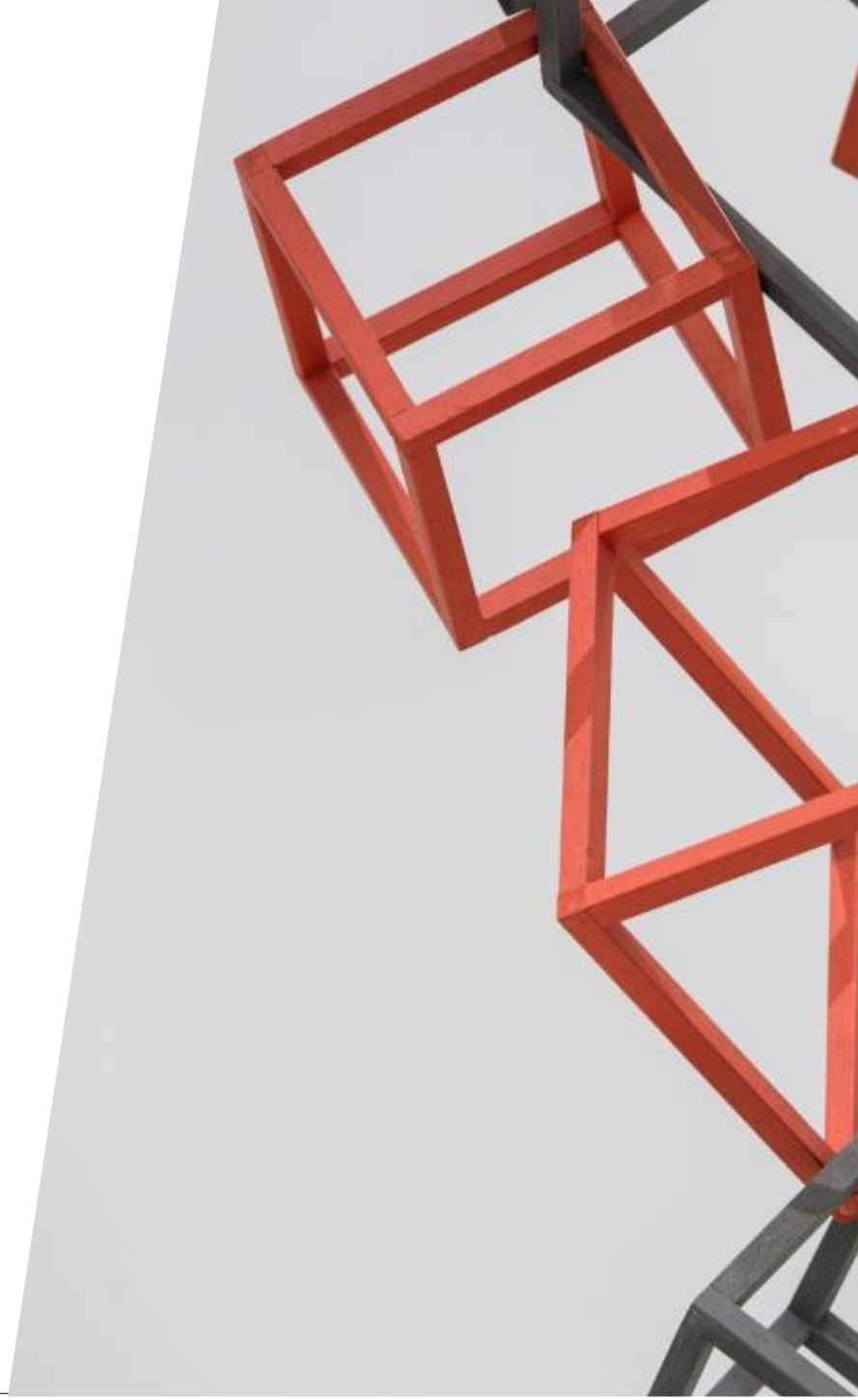
Ejercicio

- Utiliza el paquete **Beautifulsoup** para extraer datos del **Eurostoxx** de la siguiente URL: http://www.expansion.com/mercados/cotizaciones/indices/eurostoxx_I.5E.html

- Ayuda del paquete **BeautifulSoup**
  - https://pypi.python.org/pypi/beautifulsoup4
  - http://www.crummy.com/software/BeautifulSoup/
  - https://anaconda.org/conda-forge/beautifulsoup4/

- Tutorial de **BeautifulSoup**
  - http://www.analyticsvidhya.com/blog/2015/10/beginner-guide-web-scraping-beautiful-soup-python/

- *Scraping for Craft Beers: A Data Set Creation Tutorial*
  - https://www.jeannicholashould.com/python-web-scraping-tutorial-for-craft-beers.html

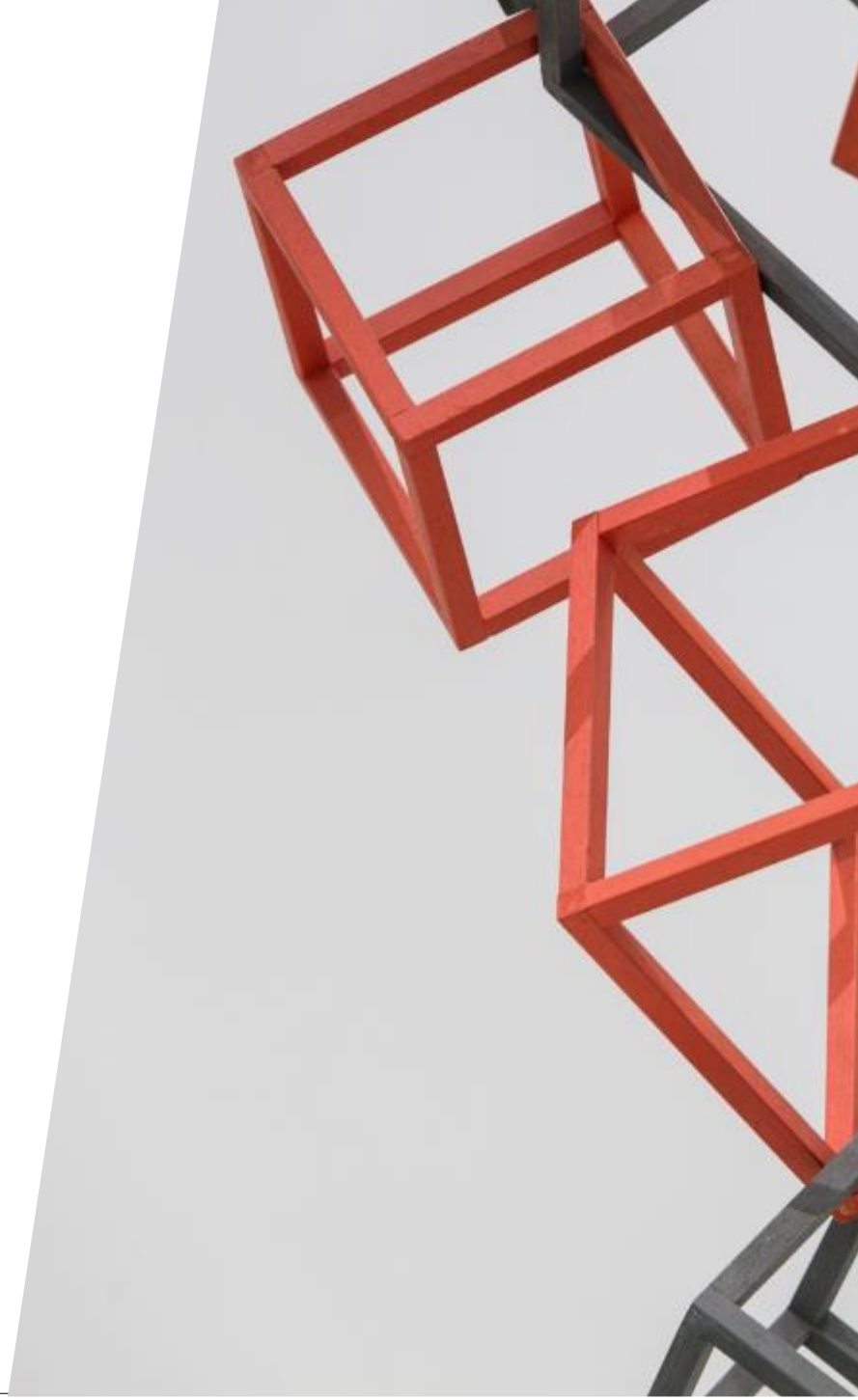


Ejercicio

- Utiliza el paquete **rvest** para extraer datos del **Eurostoxx** de la siguiente URL: http://www.expansion.com/mercados/cotizaciones/indices/eurostoxx_I.5E.html

- Ayuda del paquete **rvest**
  - https://cran.r-project.org/web/packages/rvest/rvest.pdf
  - https://github.com/tidyverse/rvest
  - https://rvest.tidyverse.org/index.html

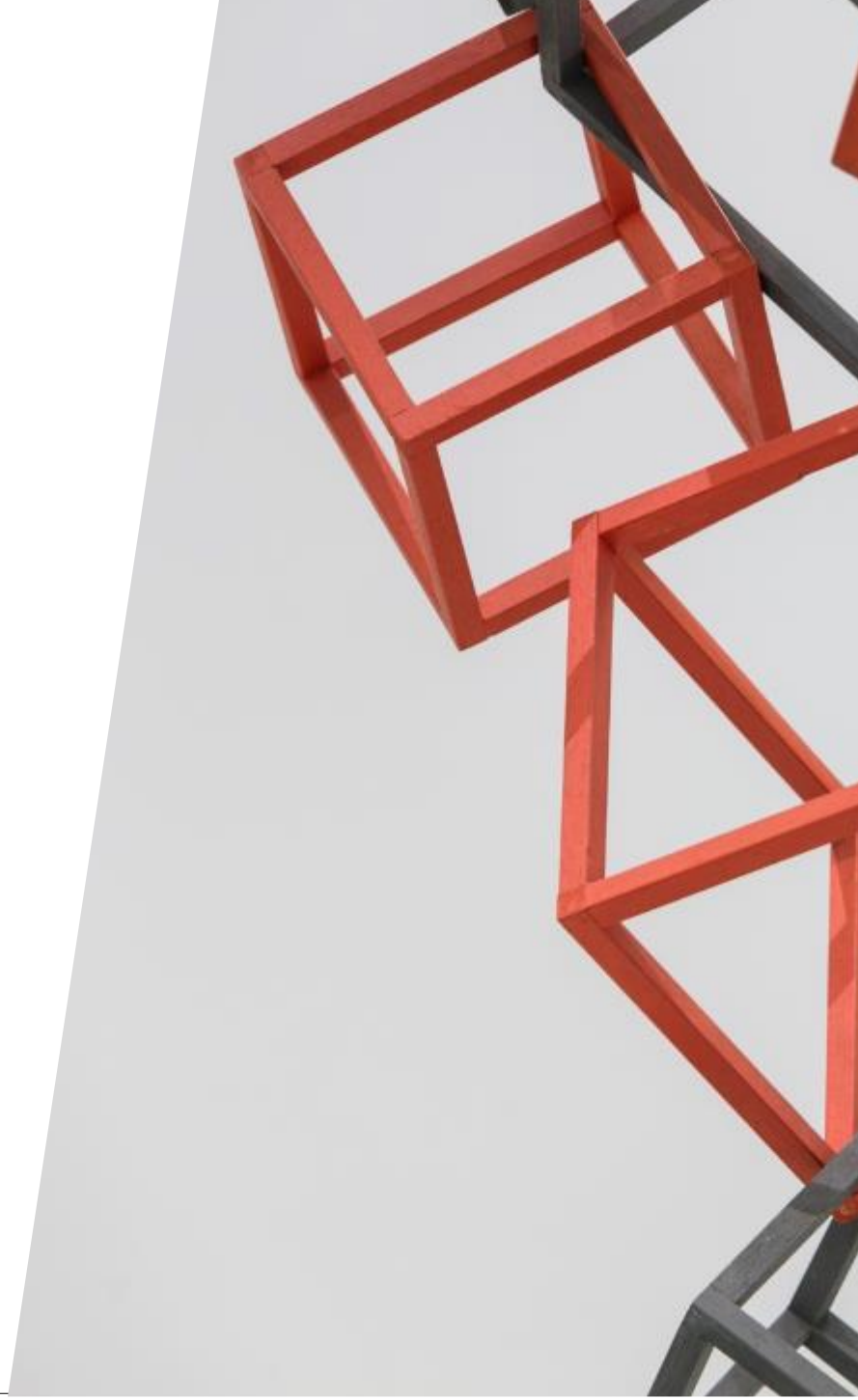
- Tutorial de **rvest**
  - https://www.r-bloggers.com/2014/11/rvest-easy-web-scraping-with-r/
  - https://www.r-bloggers.com/2020/01/web-scraping-with-rvest-astro-throwback/



Ejercicio

- Utiliza el paquete **rvest** para extraer el datos de los ratings de todos los episodios de **Juego de Tronos de IMDB** de la siguiente URL: https://www.imdb.com/title/tt0944947/

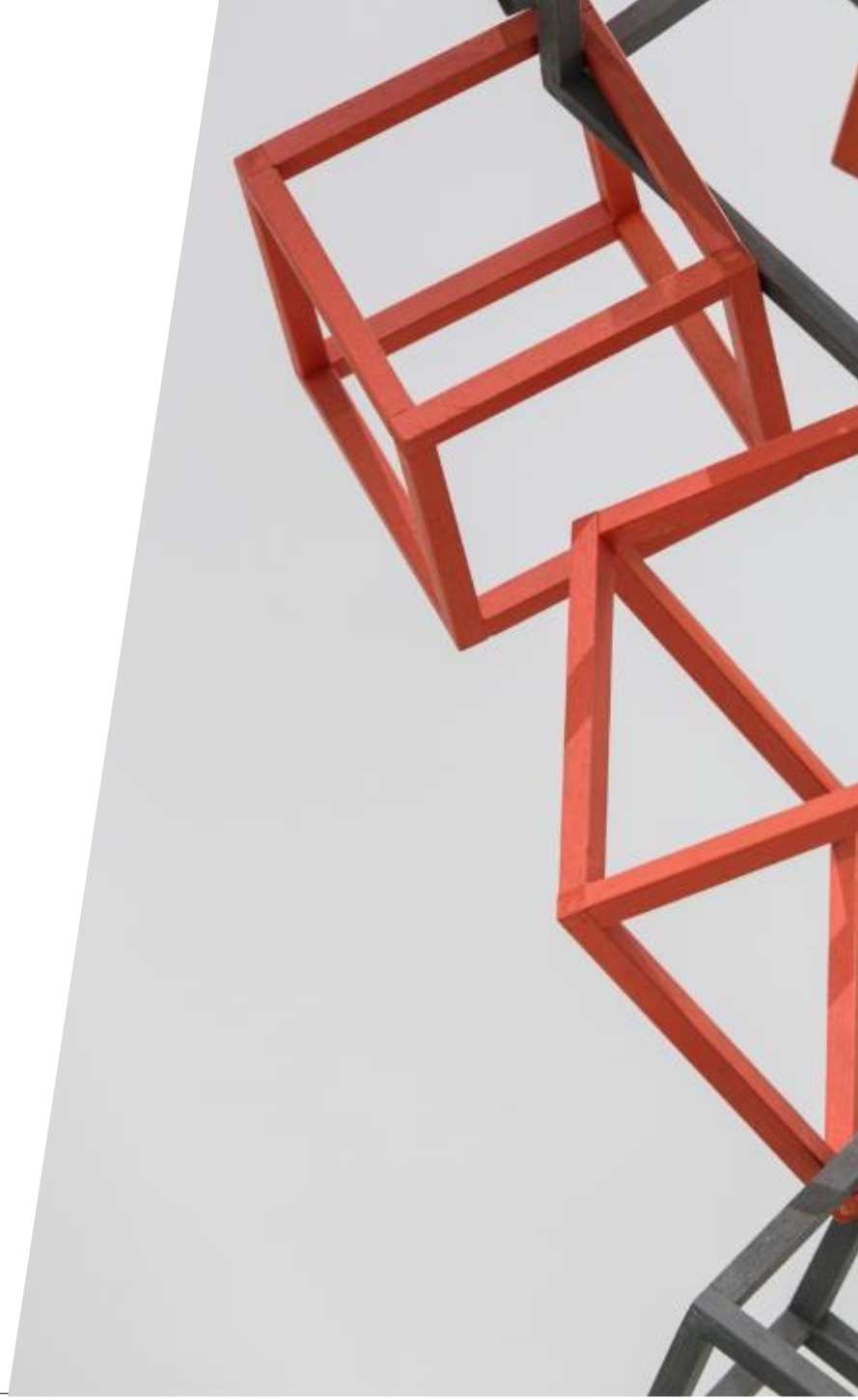
- ¿Es realmente la temporada 8 tan mala como dice la gente?



Ejercicio

- Utiliza el paquete **Scrapy** para extraer el datos de los ratings de todos los episodios de **Juego de Tronos de IMDB** de la siguiente URL: https://www.imdb.com/title/tt0944947/

- ¿Es realmente la temporada 8 tan mala como dice la gente?

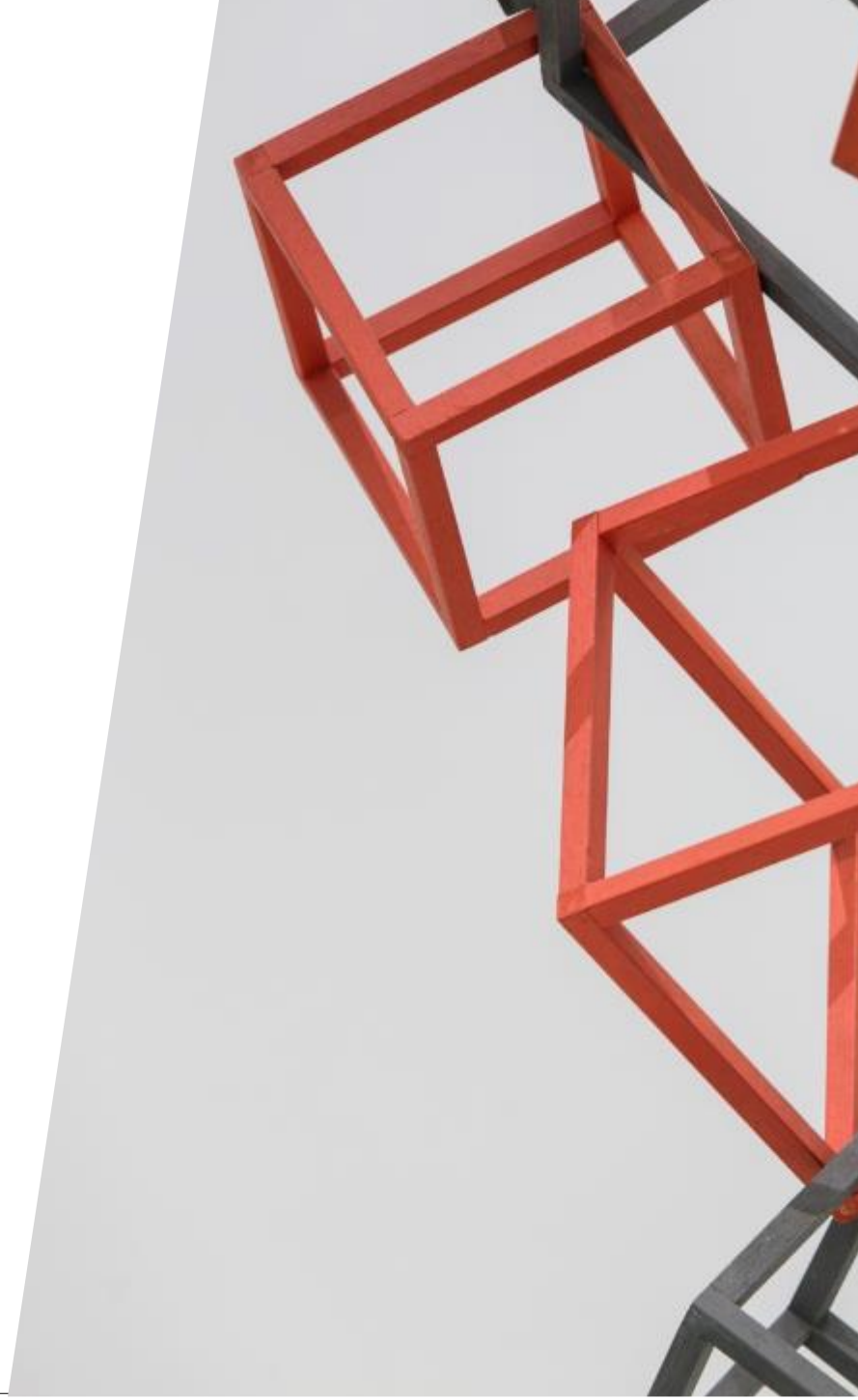


Ejercicio

- Utiliza el paquete **Scrapy** para extraer las noticias del diario **20 minutos** de la siguiente URL: https://www.20minutos.es/actualidad/

- Ayuda del paquete **Scrapy**
  - https://scrapy.org/
  - https://docs.scrapy.org/en/latest/
  - https://anaconda.org/conda-forge/scrapy

- Tutorial de **Scrapy**
  - https://www.analyticsvidhya.com/blog/2017/07/web-scraping-in-python-using-scrapy/

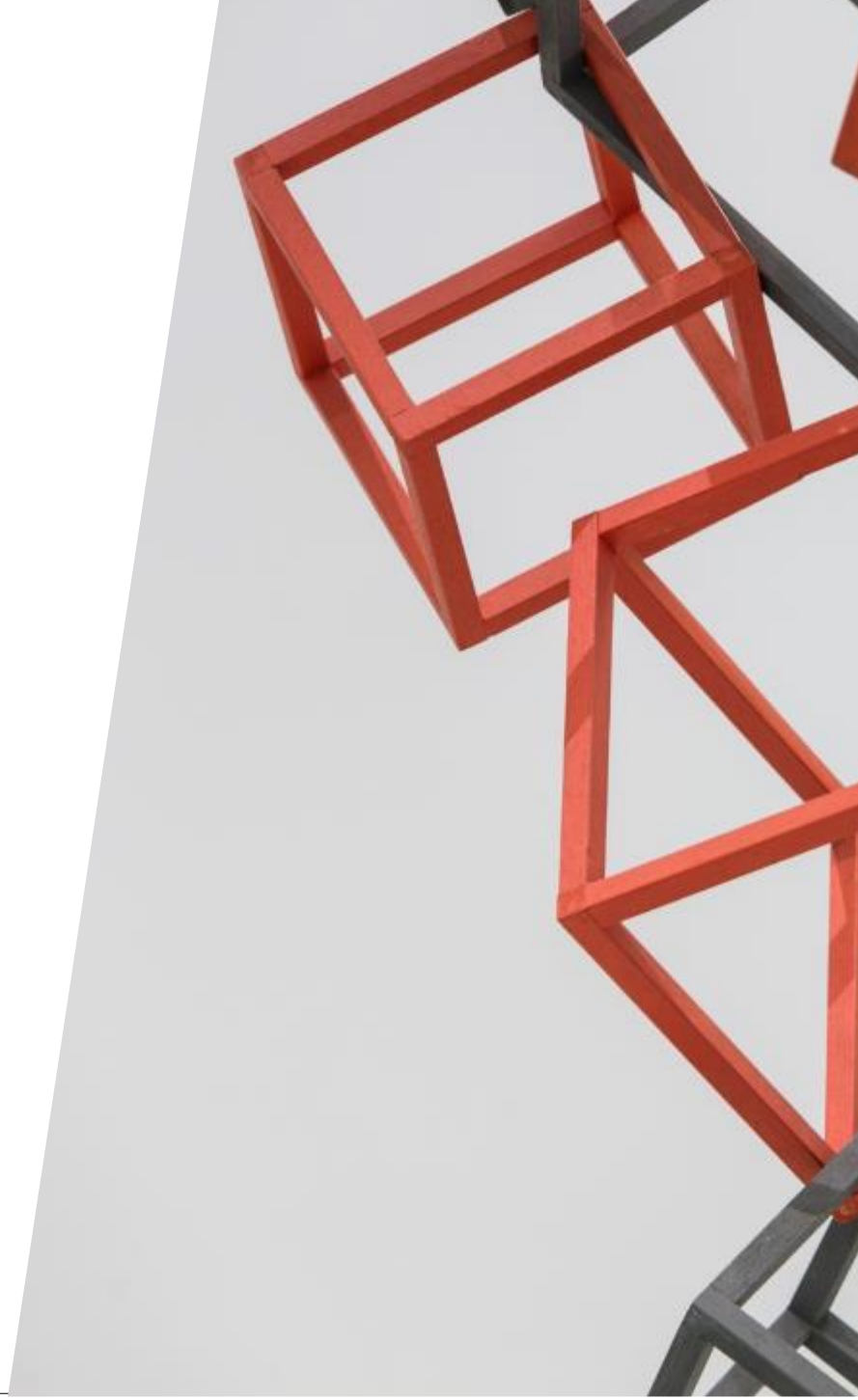


Ejercicio

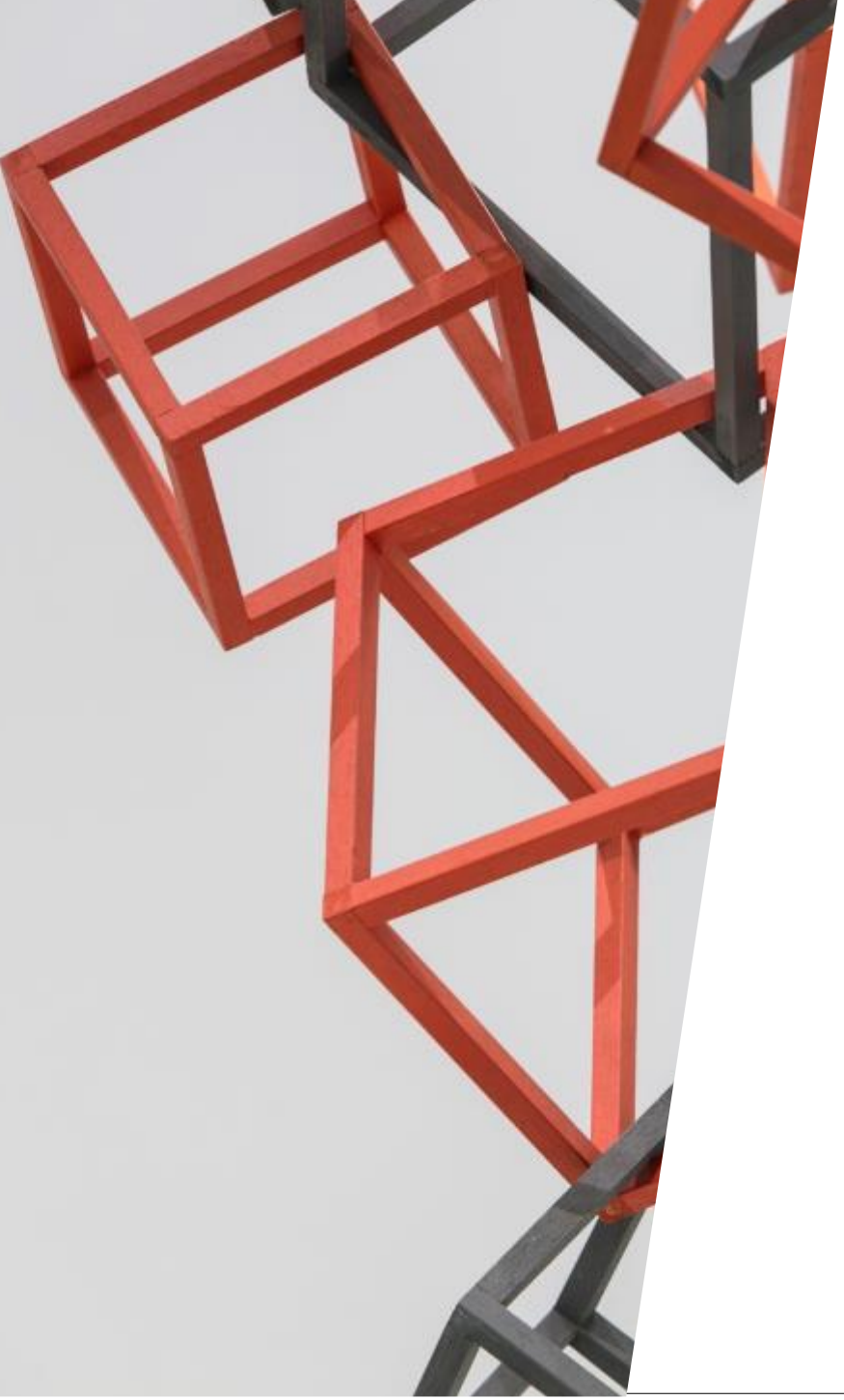
- Utiliza el paquete **Selenium** para extraer las noticias del diario **20 minutos** de la siguiente URL filtrando por un texto, sólo artículos y ordenando por fecha: https://www.20minutos.es/

- Ayuda sobre el paquete **Selenium**
  - http://selenium-python.readthedocs.io/
  - https://anaconda.org/conda-forge/selenium

- Tutorial de **Selenium**
  - https://www.analyticsvidhya.com/blog/2021/07/learn-data-scraping-using-python-and-selenium/



Web APIs



¿Qué es una web *API*?

- ***Application Programming Interface***
- Conjunto de funciones (o métodos) y procedimientos que ofrece un componente *software* (o aplicación).
- Esto permite integrar diferentes piezas de software en las aplicaciones.
- Las APIs permiten obtener datos o ejecutar una determinada tarea.
- Hablamos de APIs web cuando aplicamos este concepto a la web. La interfaz de programación es expuesta al exterior para que pueda ser empleada mediante peticiones y respuestas (mediante HTTP).

Web APIs

Elementos y partes de una *web API*.

- **URL base:** es la dirección (URL) donde se encuentra desplegada la API. Normalmente tienen la siguiente forma: https://api.twitter.com, https://api.github.com

- **Endpoint:** cada función o recurso de la API tiene una URL diferente, que empieza siempre por la URL base. Cada una de estas funciones o recursos se le conoce con el nombre de *endpoint*. La documentación oficial de la API tiene información sobre los diferentes *endpoints*.

- **API reference:** es el nombre que recibe la documentación oficial de la API. Está compuesta por la descripción de los servicios así como por ejemplos de *requests* and *responses*. También suele contar con una consola para simular peticiones a la API.

- **Requests:** peticiones/llamadas/invocaciones a la API. Contiene los parámetros, cabeceras, método usado, URL del *endpoint*, etc.

- **Responses:** información devuelta por el servidor. Incluye los datos, cabeceras, código de retorno, etc.

Web APIs

Elementos y partes de una *web API*

- **Códigos de retorno (*Status code*):** contienen información sobre el resultado de la petición y nos indican si la petición fue satisfactoria. Algunos ejemplos: 200 (OK), 400 (*Bad request*). 401 (*Unauthorized*), 404 (*Not found*), 405 (*Method not allowed*), 500 (*Internal server error*).

- **Cabeceras (*Headers*):** parámetros especiales para configurar las *requests* y *responses*. Algunos ejemplos: Accept (tipo de contenido que acepta el cliente), Content-type (tipo de contenido que devolverá el servidor), Authentication (contiene las credenciales).

- **HTTP *methods*:** también llamados verbos, especifican la acción que se desea realizar. Algunos ejemplos: POST (crear), GET (leer), PUT (actualizar), DELETE (borrar). Normalmente usaremos POST y GET para obtener los datos.

- ***Query parameters*:** colección de parámetros necesarios para realizar la acción deseada y que se pasan al *endpoint*.

Web APIs

Elementos y partes de una *web API*

- **Autenticación (*Authentication*):** sistema de identificación y de acceso a la API. Normalmente sirve de control de que se tiene permisos para usar la API. Hay diferentes métodos: API Key, OAuth, OAuth2.

- **Paginación:** en muchas ocasiones las API devuelven gran cantidad de datos. En estas ocasiones se parten los datos en porciones más pequeñas de tal modo que el servidor los puede servir de manera óptima. Se usan parámetros para establecer: resultados por página, número de página y número total de resultados.

- ***Rate limit*:** límite de peticiones (*requests*) que puede realizar un usuario en un periodo de tiempo. Por ejemplo: peticiones por segundo, peticiones al mes, etc. De este modo se controla el uso de la API, impidiendo a los usuarios hacer un mal uso de la misma.

Web APIs

Formato XML

- XML (***eXtensible Markup Languaje***) es un lenguaje de marcas empleado para almacenar datos de forma legible. Tiene una estructura jerárquica.
- Similar a HTML.
- Sirve para intercambio de información.
- Partes del documento XML:
  - **Declaración:** donde se establece la versión y la codificación del documento. También es posible declarar el tipo de documento (DTD y XML Schema).
  - **Tags:** aparecen entre los símbolos < y >.
    - *start-tags*: `<section>`
    - *end-tags*: `</section>`
    - *empty-element-tags*: `<section/>`
  - **Elementos:** componentes del documento XML. Comienzan por un *start-tag* y terminan por un *end-tag*. Entre los tags aparece el contenido del elemento (puede contener otros elementos a su vez).
  - **Atributos:** características o propiedades de los elementos (pares clave/valor). Los valores van entrecomillados (con comillas dobles).

Web APIs

Ejemplo formato XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE Edit_Mensaje SYSTEM "Edit_Mensaje.dtd">

<Edit_Mensaje>
     <Mensaje>
          <Remitente>
               <Nombre>Nombre del remitente</Nombre>
               <Mail> Correo del remitente </Mail>
          </Remitente>
          <Destinatario>
               <Nombre>Nombre del destinatario</Nombre>
               <Mail>Correo del destinatario</Mail>
          </Destinatario>
          <Texto>
               <Asunto>
                    Este es mi documento con una estructura muy sencilla
                    no contiene atributos ni entidades...
               </Asunto>
               <Parrafo>
                    Este es mi documento con una estructura muy sencilla
                    no contiene atributos ni entidades...
               </Parrafo>
          </Texto>
     </Mensaje>
</Edit_Mensaje>
```

Web APIs

Formato JSON

- JSON (***JavaScript Object Notation***) es un formato de intercambio de datos.
- Alternativa a XML gracias a su perfecta integración con JavaScript.
- Muy empleado en el desarrollo de páginas web dinámicas o SPAs (*Single Page Applications*).
- Normalmente los documentos JSON se codifican en UTF-8. Aunque también se puede emplear UTF-16 y UTF-32.
- Tipos básicos en JSON:
  - **Números:** usando punto como separador decimal.
  - **Cadenas (String):** entrecomilladas con comillas dobles.
  - **Boolean:** true o false.
  - **Array:** entre corchetes y con los valores separados por comas.
  - **null:** para representar el valor nulo.
  - **Objetos:** colecciones de pares clave/valor, separados por comas y entre llaves.

Web APIs

Ejemplo formato JSON

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

Web APIs

Comparativa entre formato XML y formato JSON

```xml
<person>
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <age>25</age>
  <address>
    <streetAddress>21 2nd Street</streetAddress>
    <city>New York</city>
    <state>NY</state>
    <postalCode>10021</postalCode>
  </address>
  <phoneNumbers>
    <phoneNumber>
      <type>home</type>
      <number>212 555-1234</number>
    </phoneNumber>
    <phoneNumber>
      <type>fax</type>
      <number>646 555-4567</number>
    </phoneNumber>
  </phoneNumbers>
  <gender>
    <type>male</type>
  </gender>
</person>
```

```json
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021"
  },
  "phoneNumber": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "fax",
      "number": "646 555-4567"
    }
  ],
  "gender": {
    "type": "male"
  }
}
```

Paquetes para trabajar con *web APIs*

Comparativa entre R y Python

R

- Peticiones HTTP: httr
- Procesamiento de formato XML: XML
- Procesamiento de formato JSON: jsonlite RJSONIO

Python

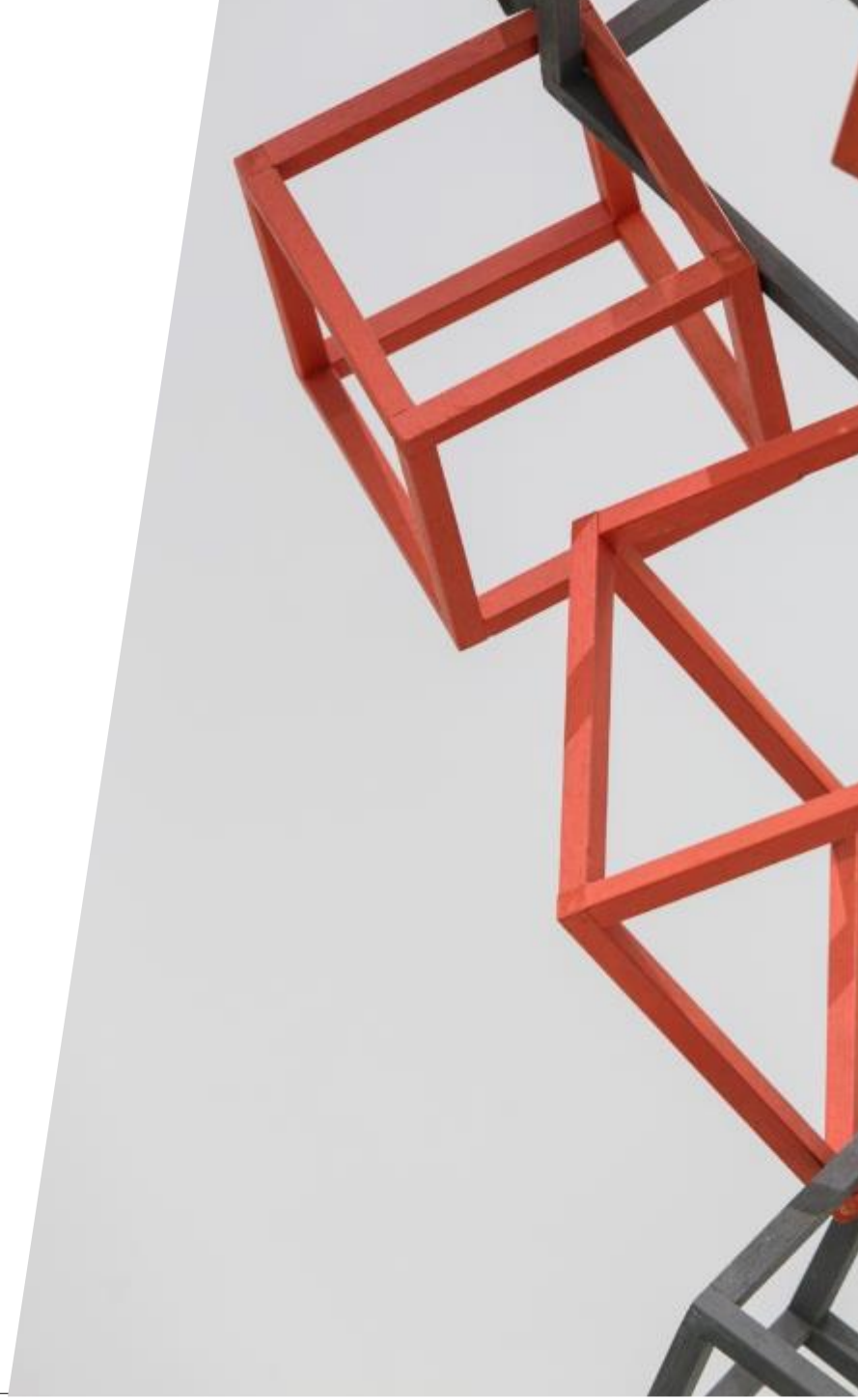
- Peticiones HTTP: requests
- Procesamiento de formato XML: lxml
- Procesamiento de formato JSON: json

Algunos detalles importantes:

- Muchas *web APIs* cuentan con paquetes desarrollados en R y/o Python que hacen que su uso sea mucho más sencillo, simplificando enormemente el proceso. Lo primero que tendremos que hacer antes de comenzar a codificar es buscar si existe un paquete que podamos usar.

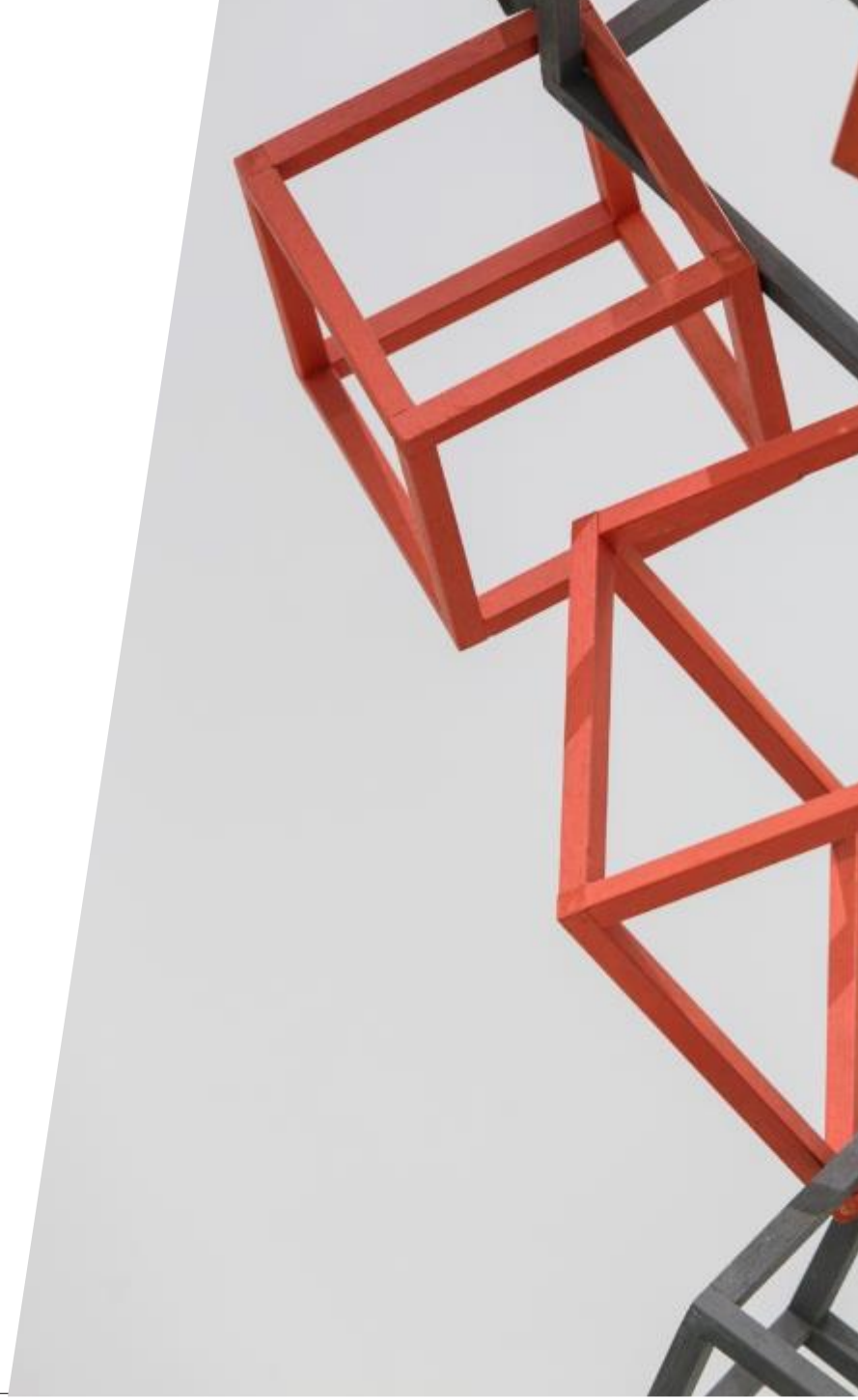
Ejercicio

- Utiliza el API de **The New York Times** y extrae los datos de los *endpoints* de **The Most Popular**:
  - Documentación general del API: http://developer.nytimes.com/
  - Documentación general de **The Most Popular**: https://developer.nytimes.com/docs/most-popular-product/1/overview
- Solicita un token, lee los datos y guárdalos en un fichero JSON
- Ayuda del paquete requests
  - https://docs.python-requests.org/en/latest/
  - https://docs.python-requests.org/en/latest/user/quickstart/
  - https://anaconda.org/conda-forge/requests
- Ayuda del paquete json
  - https://docs.python.org/3/library/json.html



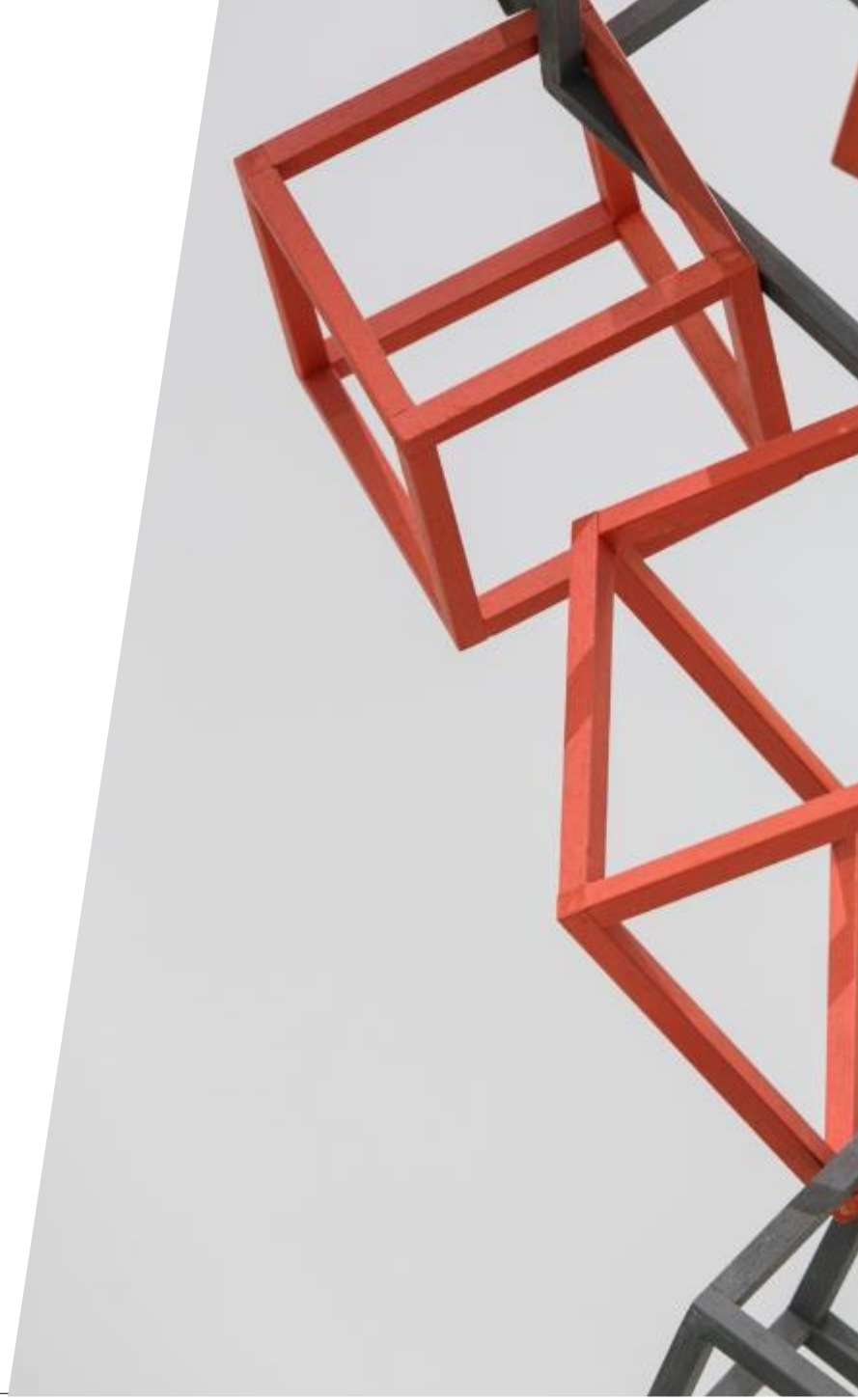
Ejercicio

- Utiliza el API de **The New York Times** y extrae los datos de los *endpoints* de **The Most Popular**:
  - Documentación general del API: http://developer.nytimes.com/
  - Documentación general de **The Most Popular**: https://developer.nytimes.com/docs/most-popular-product/1/overview
- Solicita un token, lee los datos y guárdalos en un fichero JSON
- Ayuda del paquete httr
  - https://cran.r-project.org/web/packages/httr/vignettes/quickstart.html
  - https://cran.r-project.org/web/packages/httr/httr.pdf
- Ayuda del paquete json
  - https://cran.r-project.org/web/packages/rjson/rjson.pdf



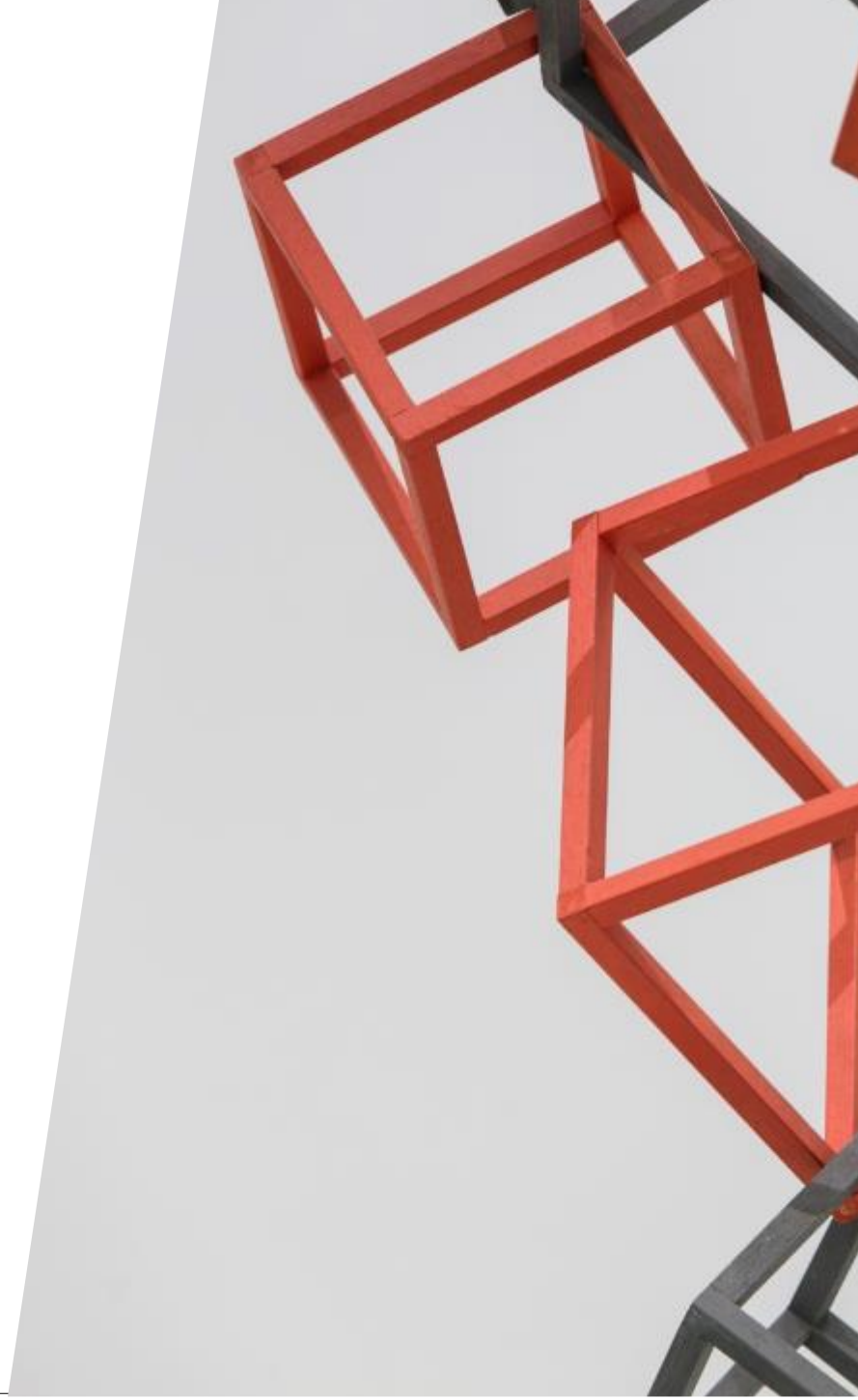
Ejercicio

- Echa un vistazo al API de **Spotify**:
  - Documentación general del API: https://developer.spotify.com/documentation/web-api/reference/
  - Documentación API de mis principales artistas y canciones: https://developer.spotify.com/documentation/web-api/reference/#endpoint-get-users-top-artists-and-tracks
  - Apps de Spotify: https://developer.spotify.com/dashboard/applications
- Solicita una API key o token
- Echa un vistazo a la documentación de spotipy:
  - Ayuda sobre el paquete spotipy: https://spotipy.readthedocs.io/en/2.18.0/
  - Autenticación: https://spotipy.readthedocs.io/en/2.18.0/#authorization-code-flow
  - Ayuda sobre el paquete json: https://docs.python.org/3/library/json.html
  - Instalación: https://anaconda.org/conda-forge/spotipy
  - Descarga tu top de canciones más escuchadas recientemente o el top de canciones más escuchadas de un determinado artista



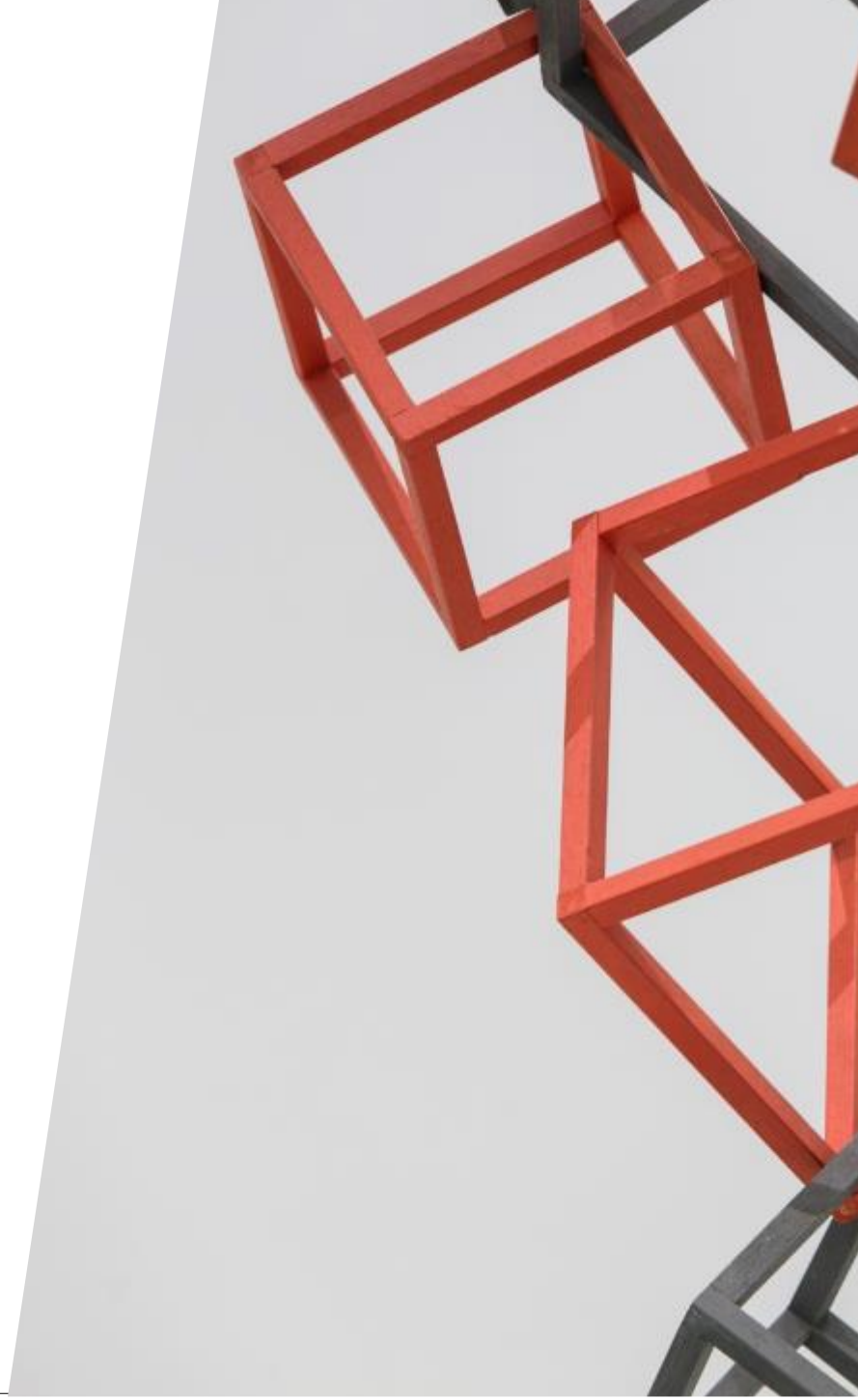
Ejercicio

- Echa un vistazo al API de **Musixmatch**:
  - Documentación API general: https://developer.musixmatch.com/documentation
  - Documentación API de letras de canciones: https://developer.musixmatch.com/documentation/api-reference/track-lyrics-get y https://developer.musixmatch.com/documentation/api-reference/matcher-lyrics-get
  - Apps de Musixmatch: https://developer.musixmatch.com/admin/applications
- Solicita una API key o token
- Echa un vistazo a la documentación de pymusixmatch:
  - Ayuda sobre el paquete pymusixmatch: https://github.com/hudsonbrendon/python-musixmatch
  - Autenticación: https://github.com/hudsonbrendon/python-musixmatch#authentication
  - Ayuda sobre el paquete json: https://docs.python.org/3/library/json.html
  - Instalación: https://github.com/hudsonbrendon/python-musixmatch#quick-start
  - Descarga las letras de tus canciones más escuchadas empleando el paquete pymusixmatch.

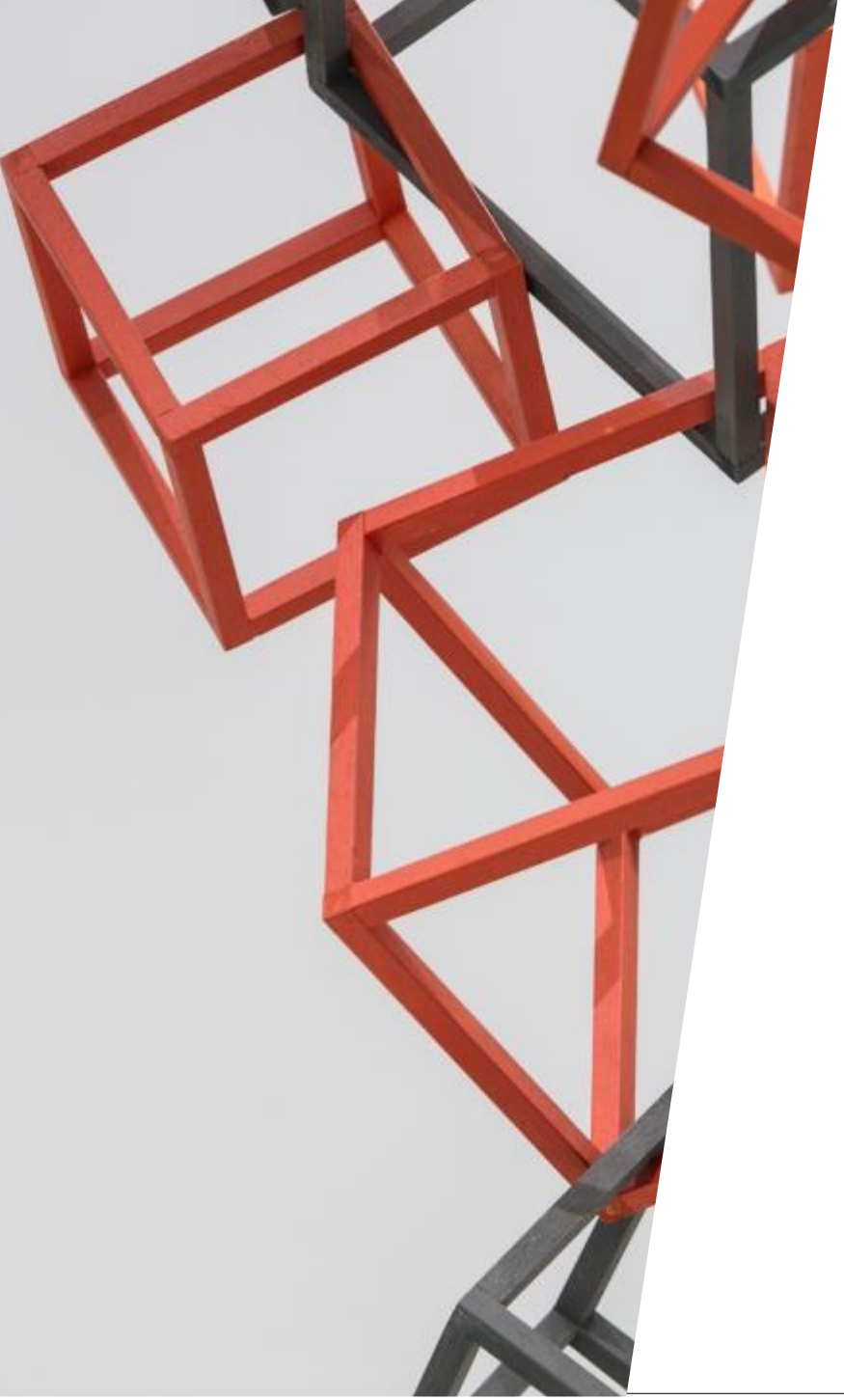


Ejercicio

- Utiliza el API de **meaningcloud**, la cual permite hacer NLP, y haz un análisis de sentimiento:
  - Documentación general de la API: https://www.meaningcloud.com/developer/apis
  - Documentación general de ***Sentiment Analysis***: https://www.meaningcloud.com/developer/sentiment-analysis
- Solicita un token y haz el análisis de sentimientos de las letras de las canciones del ejercicio anterior
- Ayuda del paquete MeaningCloud for Python
  - https://github.com/MeaningCloud/meaningcloud-python



Microdatos



¿Qué son los microdatos?

- Ficheros de datos estadísticos proporcionados por el INE (Instituto Nacional de Estadística).
- Los ficheros de microdatos contienen los datos individuales de una estadística, convenientemente anonimizados, con el fin de preservar la confidencialidad de la información.
- Los ficheros están en formato ASCII, con estructura de campos de ancho fijo y recogen para cada registro individual de la encueta los valores que toma cada variable.

Microdatos

Claves de los microdatos

- Existen ficheros de microdatos con estadísticas sobre:
  - Encuesta de Población Activa (EPA)
  - Censo de población
  - Defunciones y nacimientos
  - Encuesta de Presupuestos Familiares (EPF)
  - …

- El factor de elevación indica el número de personas de la población representadas en el registro del fichero de microdatos.

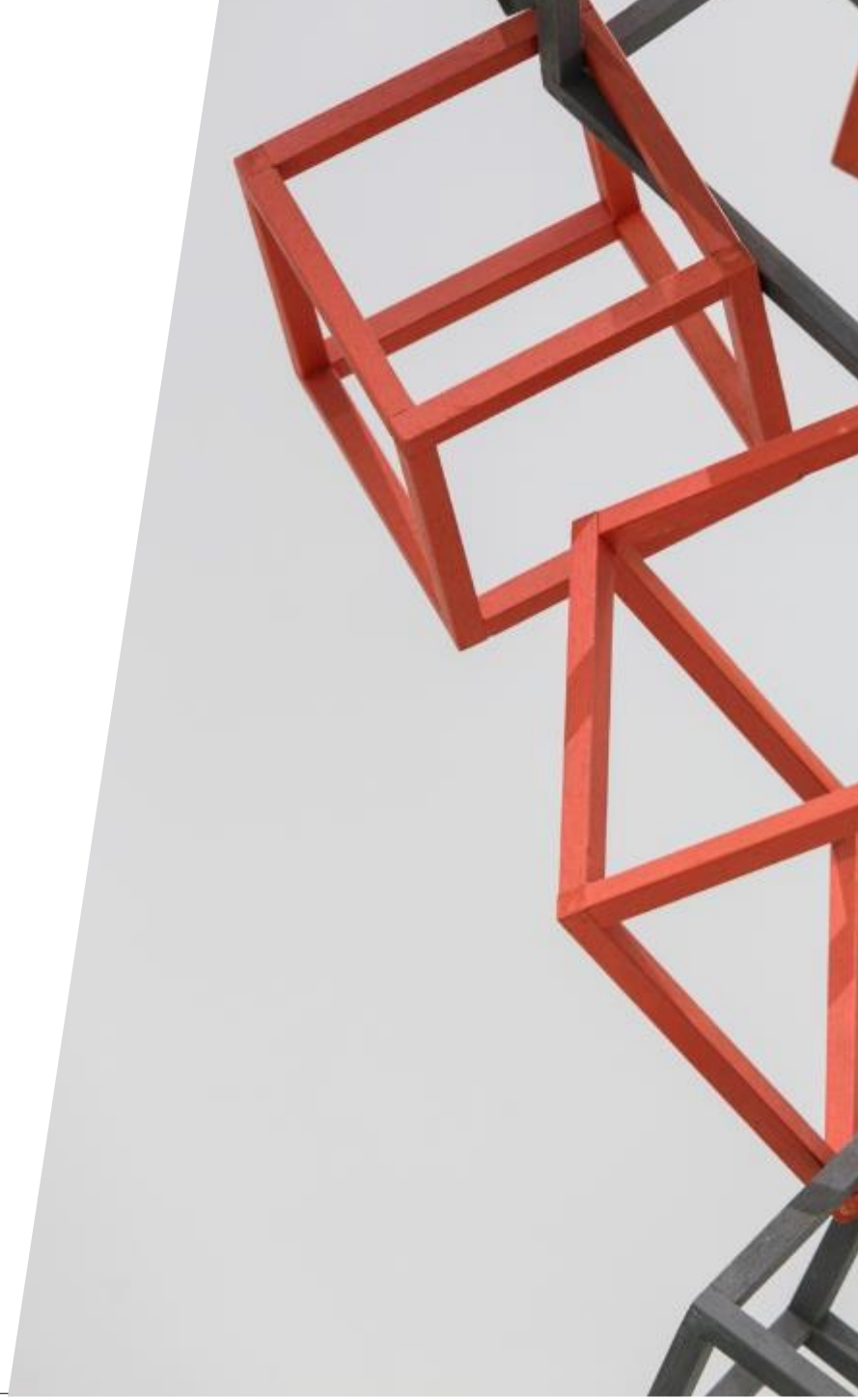
- Más información en: http://www.ine.es/prodyser/microdatos.htm

Ejercicio

- Instala el paquete MicroDatosEs de Carlos Gil Bellosta

- Calcula con el último fichero de la EPA:
  - La tasa de paro nacional
  - El número de personas ocupadas
  - El número de personas paradas
  - La tasa de actividad
  - ¿Podrías calcular la tasa de paro por sexo? ¿Por tramo de edad?
  - Puedes comprobar los resultados con la página del INE
  - Ayuda sobre el paquete MicroDatosEs: http://www.datanalytics.com/2012/08/03/el-paquete-microdataes-para-microdatos-publicos/
  - Ayuda para calcular la tasa de paro con el paquete MicroDatosEs: http://www.datanalytics.com/2012/08/06/un-paseo-por-el-paquete-microdatoses-y-la-epa-de-nuevo/
  - Documentación microdatos EPA: https://www.ine.es/ftp/microdatos/epa/dr_EPA_2021.xlsx

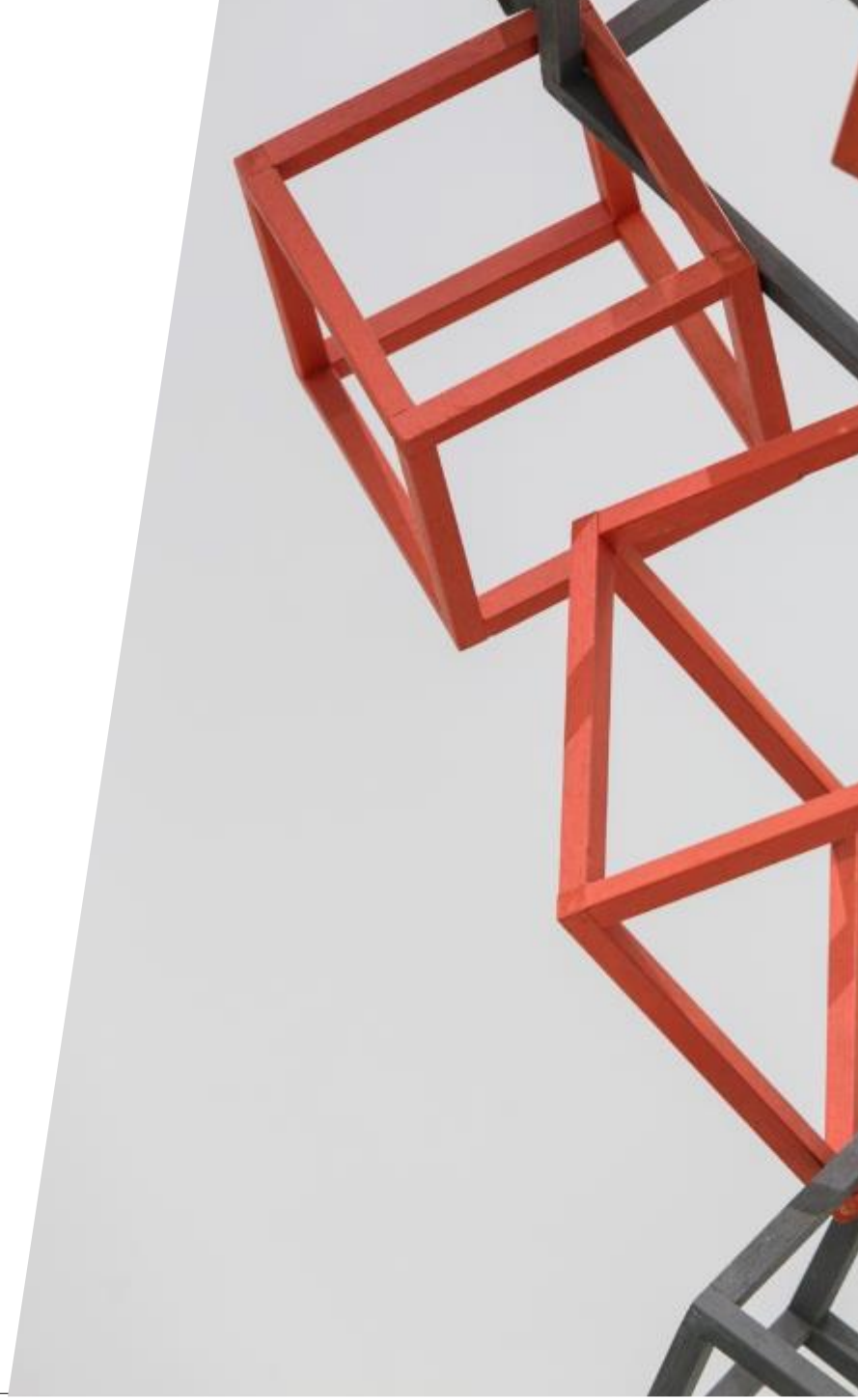
- EPA y sus limitaciones: https://escuela.civio.es/la-encuesta-de-poblacion-activa-y-sus-limitaciones/



Ejercicio

- Prueba a leer un fichero de microdatos del censo con el paquete MicroDatosEs.

- Siguiendo los pasos del ejercicio anterior crea alguna consulta de tu interés sobre las variables del fichero. Por ejemplo:
  - Porcentaje de hombres y mujeres por provincia
  - Número de mujeres con estudios universitarios por tramo de edad
  - Proporción de viudos y viudas por provincia y edad
  - …
  - Documentación microdatos censo: https://www.ine.es/ftp/microdatos/censopv/cen11/Personas%20detallado_WEB.xls



Kahoot!

Kahoot!

www.kahoot.it

Atribuciones

Atribuciones

Contenidos e imágenes

Icons made by Freepik from www.flaticon.com

Afi Escuela