

---

**CURSO:** CC50 – ADMINISTRACION DE LA INFORMACION

**CLASE:** SESION #5 (PRACTICA)

**TEMA:** ADQUISICION Y MANIPULACION DE DATOS SEMI-ESTRUCTURADOS CON R / RSTUDIO

**PROFESOR/A:** Ing. PATRICIA REYES SILVA

---

En esta clase, veremos en la práctica:

- Como adquirir datos desde formatos semi-estructurados de tipo JSON y XML.
- Repasaremos también, la adquisición de datos desde archivos de tipo CSV.
- Como leer tablas incrustadas en archivos HTML.

## **OBJETIVO PRINCIPAL**

Adquirir y manipular datos semi-estructurados en R provenientes de distintas fuentes, a fin de crear conjuntos de datos (datasets) comprensibles, en formato estructurado, antes de su pre-procesamiento.

## **COMPETENCIAS**

- Aprender a leer archivos de datos según origen, en este caso, en formato JSON, XML y CSV
- Aprender a leer tablas incrustadas en archivos HTML

## **ACTIVIDADES**

### **1. Extracción de datos utilizando URLs generadas por APIs (JSON y XML)**

- a. Extracción de datos desde la website del Instituto Nacional de Estadística de España (INE). Accederemos a ella mediante el **API JSON** que esta organización proporciona de forma libre, a fin de que cualquier persona pueda consultar la información estadística allí almacenada.
- b. Extracción de datos desde la website del Banco Mundial, la cual ofrece sus datos, también de forma libre, accediendo desde su **API XML**.

### **2. Extracción de datos desde archivos (CSV, HTML)**

- c. Extracción de datos desde archivos CSV
- d. Lectura de tablas incrustadas en archivos HTML

## **Contenido**

**I. IDENTIFICAR EL ORIGEN DE LOS DATOS**

**II. INSPECCIONAR Y CARGAR DATOS**

## I. IDENTIFICAR EL ORIGEN DE LOS DATOS

Hoy en día son populares los servicios web que proporcionan información a través de APIs que nos permiten la consulta de datos sin requerir aplicar la técnica de scraping. Estos servicios por lo general, nos devuelven los datos en formato JSON, XML o ambos (datos semi-estructurados, al igual que el formato CSV). Lo que tienen en común estos formatos es que organizan la información en forma de árbol.

Los datos a consultar provienen de las siguientes fuentes:

### a) DATOS TIPO JSON

**JSON (Java Script Object Notation)**

**Fuente:** Instituto Nacional de Estadística de España

**Consulta:** Población de cada provincia española por genero durante los últimos cinco años

[https://servicios.ine.es/wstempus/js/ES/DATOS\\_TABLA/2852?nult=5&tip=AM](https://servicios.ine.es/wstempus/js/ES/DATOS_TABLA/2852?nult=5&tip=AM)

### b) DATOS TIPO XML

**XML (Extensible Markup Language)**


**Fuente:** Banco Mundial

[https://api.worldbank.org/v2/countries/all/indicators/NY.GDP.MKTP.CD?date=2019:2020&per\\_page=500&page=1](https://api.worldbank.org/v2/countries/all/indicators/NY.GDP.MKTP.CD?date=2019:2020&per_page=500&page=1)

## II. INSPECCIONAR Y CARGAR DATOS

### a) EXTRACCIÓN DE DATOS DESDE UN JSON

El INE de España, provee una página web desde donde podemos generar las URL de los datos a consultar. Revisar: <https://www.ine.es/dyngs/DataLab/manual.html?cid=66>

  
 Instituto Nacional de Estadística

[English](#)

[Inicio](#)

[Censo Electoral](#)

[Sede electrónica](#)

[Compartir](#)

Productos y Servicios / Datos ... / API JS... / **Generador de URLs JSON**

Inicio

Definición de URLs

¿Cómo obtener identificadores de objetos utilizando INEbase?

Petición de metadatos

Petición de datos

Glosario Tempus3

**Generador de URLs JSON**

Generador de gráficos

**Generador de URLs JSON (sólo disponible para Tempus3)**

El generador permite al usuario la generación automática de URLs aplicando diferentes filtros.

▶ ¿Conoces la operación estadística sobre la que se basa la información que buscas?

¿Qué tipo de información buscas?

Datos de series

Selecciona una función

Obtener los datos de las series de una tabla de una operación estadística

Refina tu consulta:

Elige una operación estadística

Tablas de Mortalidad (Id: 197)

Selecciona una tabla

▶ **Tablas de mortalidad**

▶ Resultados nacionales, por comunidades autónomas y provincias

▶ Series desde 1991

1.1.1.1 ☐ Tablas de mortalidad por año, sexo, edad y funciones.  
 1.1.1.2 ☐ Tablas de mortalidad por año, comunidades y ciudades autónomas, sexo, edad y funciones.  
 1.1.1.3 ☒ Tablas de mortalidad por año, provincia, sexo, edad y funciones.

▶ Series 1975 - 1990

¿Cuántos datos quieres?:

☒ Cantidad de periodos:

☐ Rango de fechas (Formato: AAAAMMDD):

Generar petición

URL generada  
[https://servicios.ine.es/wstempus/js/ES/DATOS\\_TABLA/27154?nult=5](https://servicios.ine.es/wstempus/js/ES/DATOS_TABLA/27154?nult=5)

**Formato URL:**

[https://servicios.ine.es/wstempus/js/ES/DATOS\\_TABLA/{id\\_tabla}\[nult={n\\_ult\\_datos}\]\[date=AAAAMMDD:AAAAMMDD\]](https://servicios.ine.es/wstempus/js/ES/DATOS_TABLA/{id_tabla}[nult={n_ult_datos}][date=AAAAMMDD:AAAAMMDD])

**Nota:** Para evitar consultas muy grandes, el servicio web devuelve los 500 primeros ítems. Utiliza el parámetro 'page' para paginar de 500 en 500 ítems.

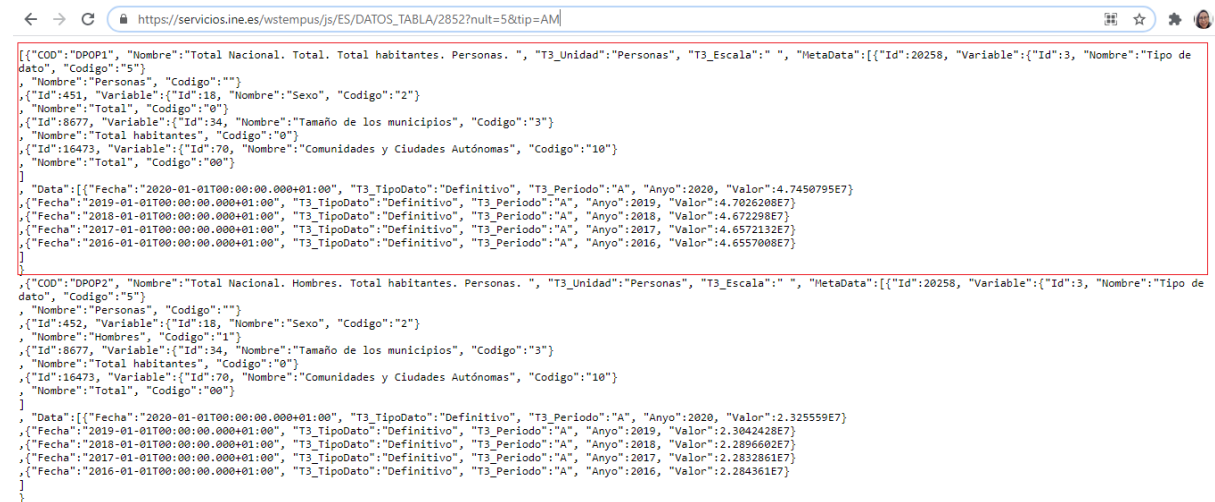
En esta página, el **INE** nos ofrece la opción de generar los URL según la consulta a datos requeridos (obviamente, necesitamos primero conocer que datos necesitamos, para luego proceder con la búsqueda).

Para efecto de esta clase práctica, trabajaremos con la URL señalada como fuente.

**Consulta:** Población de cada provincia española por genero durante los últimos cinco años

[https://servicios.ine.es/wstempus/js/ES/DATOS\\_TABLA/2852?nult=5&tip=AM](https://servicios.ine.es/wstempus/js/ES/DATOS_TABLA/2852?nult=5&tip=AM)

Entonces, al abrir este enlace obtenemos la siguiente visualización en el navegador:

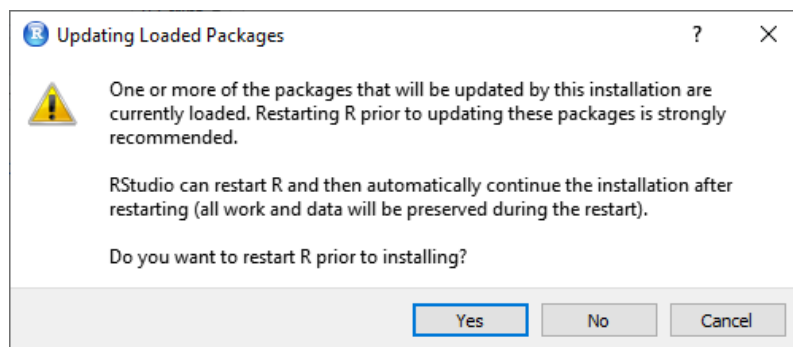


```
[{"COD": "DPO01", "Nombre": "Total Nacional. Total. Total habitantes. Personas.", "T3_Unidad": "Personas", "T3_Escala": " ", "MetaData": [{"Id": 20258, "Variable": {"Id": 3, "Nombre": "Tipo de dato", "Codigo": "5"}, "Nombre": "Personas", "Codigo": ""}, {"Id": 451, "Variable": {"Id": 18, "Nombre": "Sexo", "Codigo": "2"}, "Nombre": "Total", "Codigo": "0"}, {"Id": 8677, "Variable": {"Id": 34, "Nombre": "Tamaño de los municipios", "Codigo": "3"}, "Nombre": "Total habitantes", "Codigo": "0"}, {"Id": 16473, "Variable": {"Id": 70, "Nombre": "Comunidades y Ciudades Autónomas", "Codigo": "10"}, "Nombre": "Total", "Codigo": "00"}], "Data": [{"Fecha": "2020-01-01T00:00:00.000+01:00", "T3_TipoDato": "Definitivo", "T3_Periodo": "A", "Anyo": 2020, "Valor": 4.7450795E7}, {"Fecha": "2019-01-01T00:00:00.000+01:00", "T3_TipoDato": "Definitivo", "T3_Periodo": "A", "Anyo": 2019, "Valor": 4.7026208E7}, {"Fecha": "2018-01-01T00:00:00.000+01:00", "T3_TipoDato": "Definitivo", "T3_Periodo": "A", "Anyo": 2018, "Valor": 4.672298E7}, {"Fecha": "2017-01-01T00:00:00.000+01:00", "T3_TipoDato": "Definitivo", "T3_Periodo": "A", "Anyo": 2017, "Valor": 4.6572133E7}, {"Fecha": "2016-01-01T00:00:00.000+01:00", "T3_TipoDato": "Definitivo", "T3_Periodo": "A", "Anyo": 2016, "Valor": 4.6557008E7}], [{"COD": "DPO02", "Nombre": "Total Nacional. Hombres. Total habitantes. Personas.", "T3_Unidad": "Personas", "T3_Escala": " ", "MetaData": [{"Id": 20258, "Variable": {"Id": 3, "Nombre": "Tipo de dato", "Codigo": "5"}, "Nombre": "Personas", "Codigo": ""}, {"Id": 452, "Variable": {"Id": 18, "Nombre": "Sexo", "Codigo": "2"}, "Nombre": "Hombres", "Codigo": "1"}, {"Id": 8677, "Variable": {"Id": 34, "Nombre": "Tamaño de los municipios", "Codigo": "3"}, "Nombre": "Total habitantes", "Codigo": "0"}, {"Id": 16473, "Variable": {"Id": 70, "Nombre": "Comunidades y Ciudades Autónomas", "Codigo": "10"}, "Nombre": "Total", "Codigo": "00"}], "Data": [{"Fecha": "2020-01-01T00:00:00.000+01:00", "T3_TipoDato": "Definitivo", "T3_Periodo": "A", "Anyo": 2020, "Valor": 2.325559E7}, {"Fecha": "2019-01-01T00:00:00.000+01:00", "T3_TipoDato": "Definitivo", "T3_Periodo": "A", "Anyo": 2019, "Valor": 2.3042428E7}, {"Fecha": "2018-01-01T00:00:00.000+01:00", "T3_TipoDato": "Definitivo", "T3_Periodo": "A", "Anyo": 2018, "Valor": 2.2896602E7}, {"Fecha": "2017-01-01T00:00:00.000+01:00", "T3_TipoDato": "Definitivo", "T3_Periodo": "A", "Anyo": 2017, "Valor": 2.2832861E7}, {"Fecha": "2016-01-01T00:00:00.000+01:00", "T3_TipoDato": "Definitivo", "T3_Periodo": "A", "Anyo": 2016, "Valor": 2.284361E7}]}]
```

Observamos que los bloques de datos se repiten, y estos tienen una estructura dividida en dos partes: “COD” y “Data”. Estas partes o bloques principales están a compuestos por otros datos (listas de datos).

**En R/RStudio ejecutaremos las siguientes instrucciones.**

- Creamos un archivo R en nuestro directorio de trabajo, al que llamaremos 01-analisis-json
- Trabajaremos con el paquete **jsonlite**, el que instalaremos con la instrucción `install.packages("jsonlite")`. Este paquete, nos simplificara el acceso a los datos, en lugar de simplemente utilizar el paquete `rjson`.
- Como primera línea del script colocamos `install.packages("jsonlite")` y ejecutamos con `Ctrl+Enter` nos pedirá reiniciar R porque deberá actualizar otros paquetes (clic en **Yes** y veremos el resultado de la instalación en la consola)



- Cargamos la librería con `library(jsonlite)`
- Creamos la variable `ine.url` y le asignamos la URL JSON a consultar

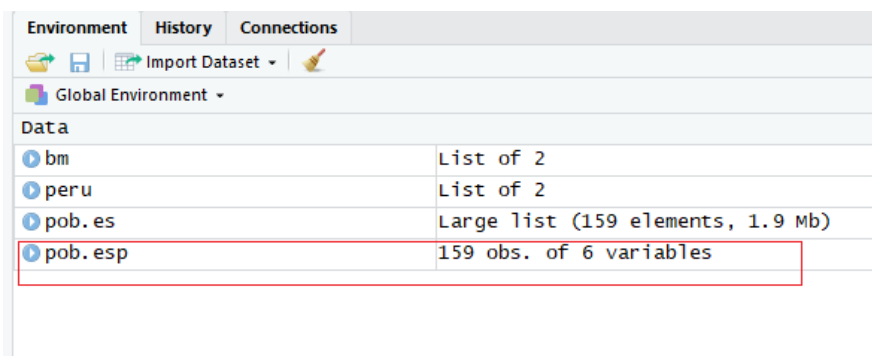
```
ine.url <- "https://servicios.ine.es/wstempus/js/ES/DATOS_TABLA/2852?nult=5&tip=AM"
```

- Utilizamos la instrucción `fromJSON()` para cargar los datos contenido en `ine.url` en una variable de datos llamada `pob.esp`

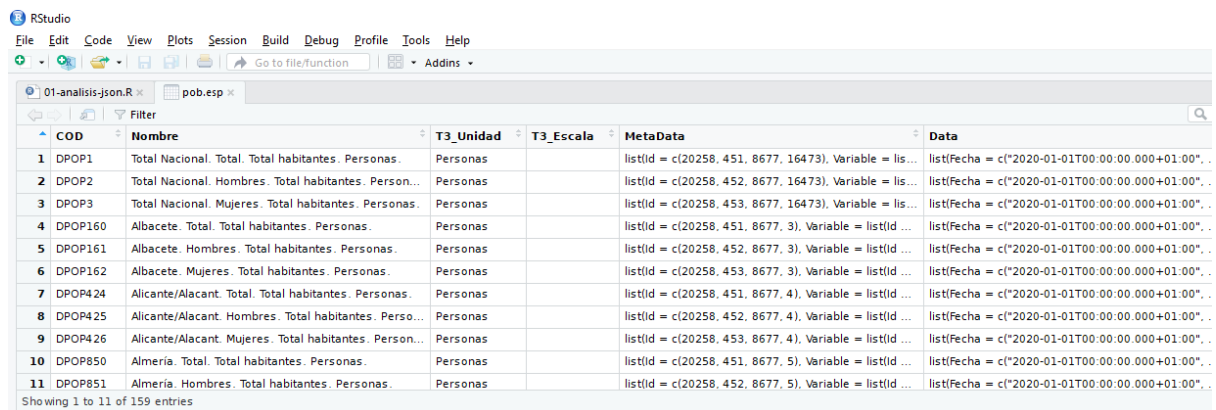
```
pob.esp <- fromJSON(ine.url)
```

Damos Ctrl+Enter (cada vez que se quiere ejecutar una línea de código dentro del script).

- En el área/ambiente global en RStudio verificamos que se creó la variable `pob.esp` y contiene 159 observaciones en 6 variables.



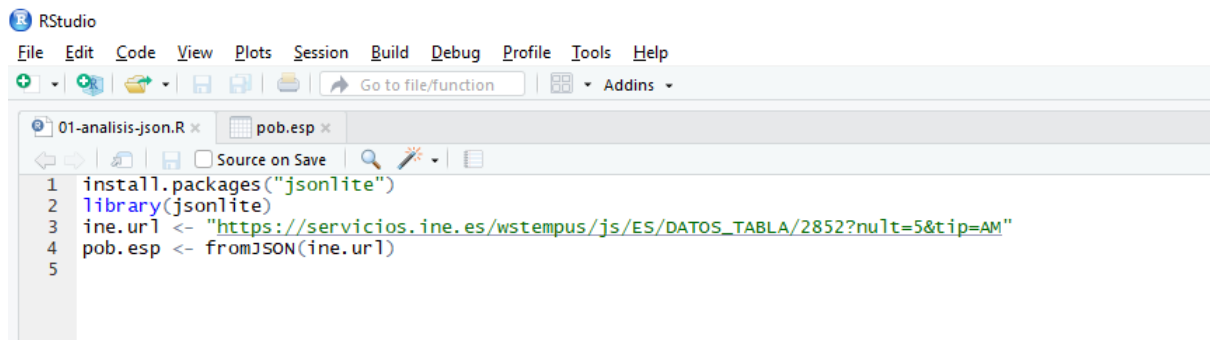
Damos clic sobre dichos datos y visualizaremos el contenido en formato de tabla:



COD	Nombre	T3_Unidad	T3_Escala	MetaData	Data
1 DPOP1	Total Nacional. Total. Total habitantes. Personas.	Personas		list(lid = c(20258, 451, 8677, 16473), Variable = lis...	list(Feche = c("2020-01-01T00:00:00.000+01:00", ...
2 DPOP2	Total Nacional. Hombres. Total habitantes. Person...	Personas		list(lid = c(20258, 452, 8677, 16473), Variable = lis...	list(Feche = c("2020-01-01T00:00:00.000+01:00", ...
3 DPOP3	Total Nacional. Mujeres. Total habitantes. Personas.	Personas		list(lid = c(20258, 453, 8677, 16473), Variable = lis...	list(Feche = c("2020-01-01T00:00:00.000+01:00", ...
4 DPOP160	Albacete. Total. Total habitantes. Personas.	Personas		list(lid = c(20258, 451, 8677, 3), Variable = list(lid ...	list(Feche = c("2020-01-01T00:00:00.000+01:00", ...
5 DPOP161	Albacete. Hombres. Total habitantes. Personas.	Personas		list(lid = c(20258, 452, 8677, 3), Variable = list(lid ...	list(Feche = c("2020-01-01T00:00:00.000+01:00", ...
6 DPOP162	Albacete. Mujeres. Total habitantes. Personas.	Personas		list(lid = c(20258, 453, 8677, 3), Variable = list(lid ...	list(Feche = c("2020-01-01T00:00:00.000+01:00", ...
7 DPOP424	Alicante/Alacant. Total. Total habitantes. Personas.	Personas		list(lid = c(20258, 451, 8677, 4), Variable = list(lid ...	list(Feche = c("2020-01-01T00:00:00.000+01:00", ...
8 DPOP425	Alicante/Alacant. Hombres. Total habitantes. Perso...	Personas		list(lid = c(20258, 452, 8677, 4), Variable = list(lid ...	list(Feche = c("2020-01-01T00:00:00.000+01:00", ...
9 DPOP426	Alicante/Alacant. Mujeres. Total habitantes. Person...	Personas		list(lid = c(20258, 453, 8677, 4), Variable = list(lid ...	list(Feche = c("2020-01-01T00:00:00.000+01:00", ...
10 DPOP850	Almeria. Total. Total habitantes. Personas.	Personas		list(lid = c(20258, 451, 8677, 5), Variable = list(lid ...	list(Feche = c("2020-01-01T00:00:00.000+01:00", ...
11 DPOP851	Almeria. Hombres. Total habitantes. Personas.	Personas		list(lid = c(20258, 452, 8677, 5), Variable = list(lid ...	list(Feche = c("2020-01-01T00:00:00.000+01:00", ...

**Nota:** la instrucción `fromJSON()` puede recibir bien una URL cuyo resultado sean datos de tipo JSON o bien, el nombre de un archivo con extensión `.json` (entre comillas dobles y que se encuentre en nuestra PC o en Internet), los trata de igual forma.

El script en R debió haber quedado de esta forma:



```
1 install.packages("jsonlite")
2 library(jsonlite)
3 ine.url <- "https://servicios.ine.es/wstempus/js/ES/DATOS_TABLA/2852?nult=5&tip=AM"
4 pob.esp <- fromJSON(ine.url)
5
```

### Ejercicio:

Realizar el mismo ejercicio, para:

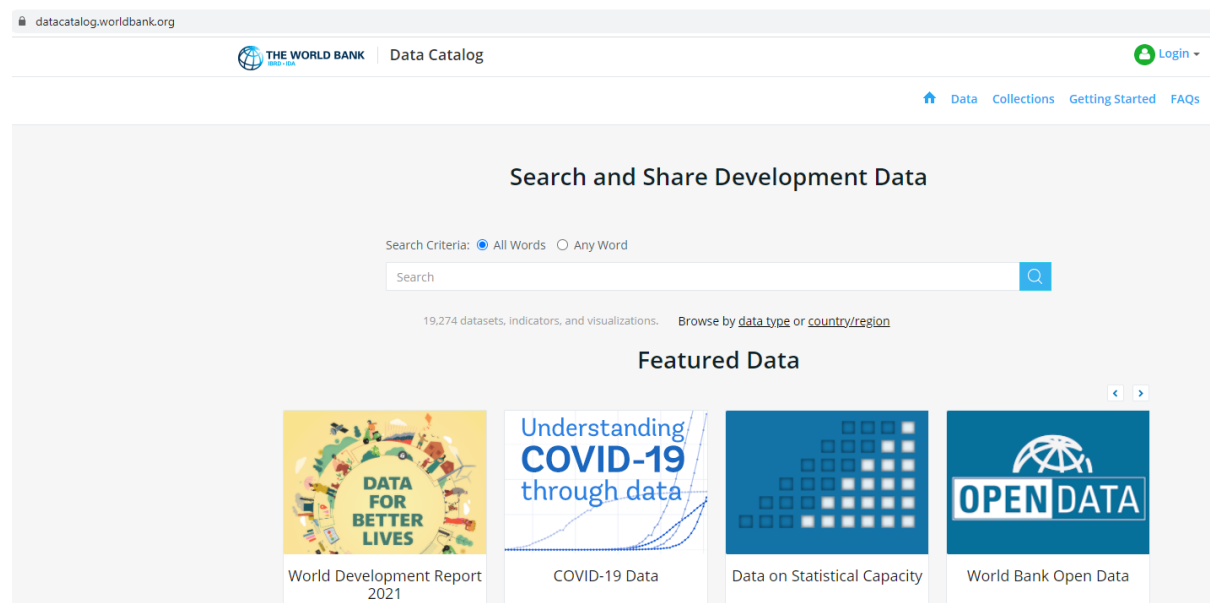
- (1) La URL: <https://www.floatrates.com/daily/usd.json>
- (2) Los archivos .json adjuntos en el aula virtual llamados: students.json y student-courses.json

Guardar los scripts con los nombres: 01-analisis-json-url y 01-analisis-json-files respectivamente.

### b) EXTRACCIÓN DE DATOS DESDE UN XML


El Banco Mundial ofrece mediante servicios web de acceso a sus datos, sea mediante descarga manual o vía el uso de su API.

Podemos acceder a la descarga de los datos libres que ofrece el Banco Mundial desde este enlace: <https://datacatalog.worldbank.org/>



Y para ayudarnos a conocer las estructuras de llamadas básicas a su API visitamos este enlace: <https://datahelpdesk.worldbank.org/knowledgebase/articles/898581-api-basic-call-structures>

datahelpdesk.worldbank.org/knowledgebase/articles/898581-api-basic-call-structures


**THE WORLD BANK** | Working for a World Free of Poverty

inglés Español Français العربية Русский 中文

Casa Acerca de **Datos** Investigar Aprendiendo Noticias Proyectos y Operaciones Publicaciones Países Temas

## Datos

### Estructuras de llamadas básicas de API

— Información para desarrolladores

La API de indicadores admite dos formas básicas de crear consultas: una estructura basada en URL y una estructura basada en argumentos. Por ejemplo, las dos solicitudes siguientes devolverán los mismos datos, una lista de países con un nivel de ingresos clasificado como de ingresos bajos:

- Basado en argumentos: <http://api.worldbank.org/v2/country?incomeLevel=LIC>
- Basado en URL: <http://api.worldbank.org/v2/incomeLevel/LIC/country>

### Cadenas de consulta

La API admite las siguientes cadenas de consulta en las solicitudes.

**Fecha y intervalo de fechas :** intervalo de fechas por año, mes o trimestre que abarca el conjunto de resultados.

Ejemplos:

- <http://api.worldbank.org/v2/country/all/indicator/SP.POP.TOTL?date=2000>
- <http://api.worldbank.org/v2/country/chn/bra/indicator/DPANUSSPB?date=2012M01>

Un rango se indica mediante el **:** separador de dos puntos ( ).

Ejemplos:

- <http://api.worldbank.org/v2/country/all/indicator/SP.POP.TOTL?date=2000-2001>
- <http://api.worldbank.org/v2/country/chn/bra/indicator/DPANUSSPB?date=2012M01-2012M08>
- <http://api.worldbank.org/v2/country/CHL/indicator/DP.DOD.DECD.CR.BC.CD?date=2013Q1-2013Q4>

Las solicitudes también admiten valores del año hasta la fecha (YTD), lo que es útil para consultar datos de alta frecuencia.

- Ejemplo: <http://api.worldbank.org/v2/country/chn/indicator/DPANUSSPB?date=YTD-2013>

**Formato de salida:** la API admite los siguientes cuatro formatos de salida.

- Formato XML: <http://api.worldbank.org/v2/country/all/indicator/SP.POP.TOTL?format=xml>
- Formato JSON: <http://api.worldbank.org/v2/country/all/indicator/SP.POP.TOTL?format=json>
- Formato JSONP: <http://api.worldbank.org/v2/country/all/indicator/SP.POP.TOTL?format=jsonp&prefix=Getdata>
- Nota: Para el formato JSONP, *prefix* se debe especificar el parámetro.
- Formato JSON-stat: <http://api.worldbank.org/v2/country/all/indicator/SP.POP.TOTL?format=jsonstat>
- Nota: Consulte <https://json-stat.org/> para obtener más detalles.

**Formato de descarga:** la API admite los siguientes tres formatos de descarga.

- Descarga CSV (Descargas a archivo ZIP): <http://api.worldbank.org/v2/country/ind/indicator/AG.AGR.TRAC.NO?source=2&downloadformat=csv>
- Descarga XML (Descargas a archivo ZIP):

Los usuarios nuevos y recurrentes pueden **iniciar sesión**

Gracias por visitar la mesa de ayuda de datos del Banco Mundial. Revise las condiciones de uso de este sitio web. Su uso continuado de este sitio web constituye su aceptación de estos términos y condiciones.

**Información del desarrollador**

**Soporte de contacto**

**Dar opinión**

Sugerencias generales **0**

Datos nuevos **0**

Mejoras del sitio web **0**

**Base de conocimientos**

Clasificación de país **6**

Monedas **4**

Metodología de recopilación de datos **11**

Datos no disponibles **7**

Actualizaciones de datos **3**

DataBank **13**

Información para desarrolladores **16**

Deuda externa **5**

Finanzas / Préstamos **1**

Inversión extranjera directa (IED) **2**

Producto interno bruto (PIB) / Ingreso nacional bruto (INB) **4**

PCI / APP **1**

Programa de comparación internacional (PCI) **2**

Microdatos **15**

Entrenamiento de datos abiertos **1**

Existen innumerables datasets y los formatos de descarga/acceso a los datos es variado (CSV, JSON y XML como los principales).

Para efecto de esta clase práctica, trabajaremos con la URL señalada como fuente.

**Consulta:** Banco Mundial

Muestra en formato XML que extrae el PIB de 500 países entre para los años 2019 y 2020 (para el API del Banco Mundial, el indicador PIB se denomina NY.GDP.MKTP (según su documentación).

[http://api.worldbank.org/v2/countries/all/indicators/NY.GDP.MKTP.CD?date=2019:2020&per\\_page=500&page=1](http://api.worldbank.org/v2/countries/all/indicators/NY.GDP.MKTP.CD?date=2019:2020&per_page=500&page=1)

Entonces, al abrir este enlace obtenemos la siguiente visualización en el navegador:

← → [https://api.worldbank.org/v2/countries/all/indicators/NY.GDP.MKTP.CD?date=2019:2020&per\\_page=500&page=1](https://api.worldbank.org/v2/countries/all/indicators/NY.GDP.MKTP.CD?date=2019:2020&per_page=500&page=1)

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <data>
    <indicator id="NY.GDP.MKTP.CD">GDP (current US$)</indicator>
    <country id="1A">Arab World</country>
    <countryiso3code>ARB</countryiso3code>
    <date>2020</date>
    <value/>
    <unit/>
    <obs_status/>
    <decimal>0</decimal>
  </data>
  <data>
    <indicator id="NY.GDP.MKTP.CD">GDP (current US$)</indicator>
    <country id="1A">Arab World</country>
    <countryiso3code>ARB</countryiso3code>
    <date>2019</date>
    <value>2817414584665.11</value>
    <unit/>
    <obs_status/>
    <decimal>0</decimal>
  </data>
  <data>
    <indicator id="NY.GDP.MKTP.CD">GDP (current US$)</indicator>
    <country id="S3">Caribbean small states</country>
    <countryiso3code>CSS</countryiso3code>
    <date>2020</date>
    <value/>
    <unit/>
    <obs_status/>
    <decimal>0</decimal>
  </data>
  <data>
    <indicator id="NY.GDP.MKTP.CD">GDP (current US$)</indicator>
    <country id="S3">Caribbean small states</country>
    <countryiso3code>CSS</countryiso3code>
    <date>2019</date>
    <value>77721714917.8506</value>
    <unit/>
    <obs_status/>
    <decimal>0</decimal>
  </data>
  <data>
    <indicator id="NY.GDP.MKTP.CD">GDP (current US$)</indicator>
    <country id="B8">Central Europe and the Baltics</country>
    <countryiso3code>CEB</countryiso3code>
    <date>2020</date>
    <value/>
    <unit/>
    <obs_status/>
    <decimal>0</decimal>
  </data>
  <data>
    <indicator id="NY.GDP.MKTP.CD">GDP (current US$)</indicator>
    <country id="B8">Central Europe and the Baltics</country>
    <countryiso3code>CEB</countryiso3code>
    <date>2019</date>
    <value>1668851928931.89</value>
    <unit/>
    <obs_status/>
    <decimal>0</decimal>
  </data>
</root>
```

## En R/RStudio ejecutaremos las siguientes instrucciones.

- Creamos un archivo R en nuestro directorio de trabajo, al que llamaremos 01-analisis-xml
- Para trabajar con datos de tipo XML en R se necesita un paquete adicional: XML, el que instalaremos con la instrucción `install.packages("XML")`.
- Como primera línea del script colocamos `install.packages("XML")` y ejecutamos con Ctrl+Enter (verificamos que en la consola se haya instalado el paquete correctamente).
- Cargamos la librería con `library(XML)`
- Creamos la variable string `bm.url` y le asignamos la URL XML a consultar

`bm.url <-`

`"http://api.worldbank.org/v2/countries/all/indicators/NY.GDP.MKTP.CD?date=2019:2020&per_page=500&page=1"`

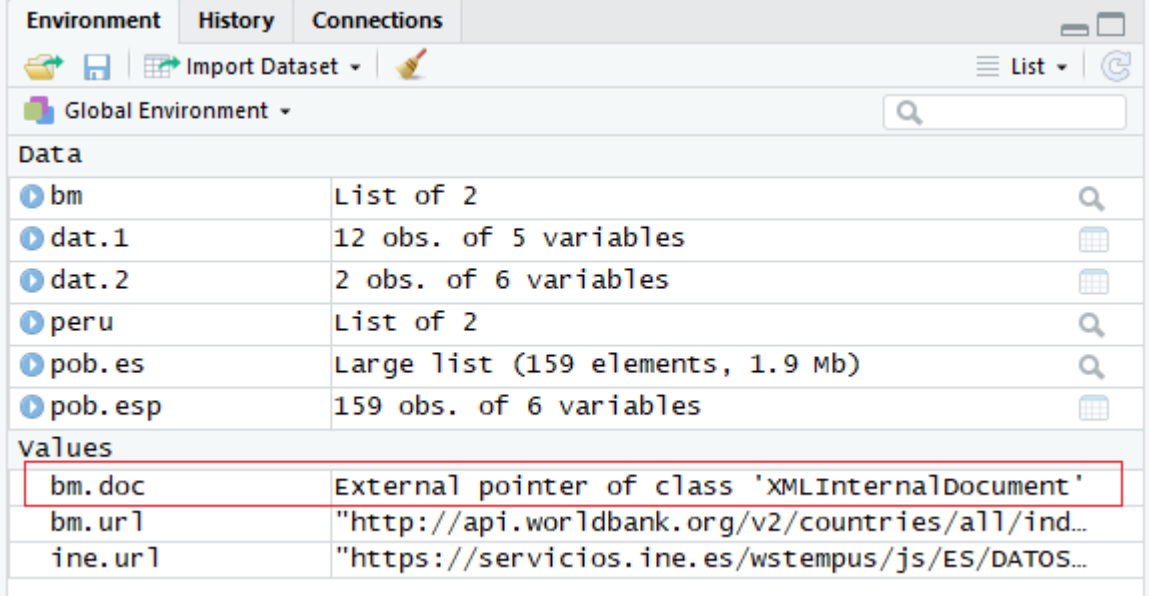
- Utilizamos la instrucción `xmlParse()`, que lo que hace es crear un puntero hacia el documento `bm.doc <- xmlParse(bm.url)`

Damos Ctrl+Enter (cada vez que se quiere ejecutar una línea de código dentro del script).

**Nota curiosa:** si utilizamos la url https, pues no reconocerá el contenido como de formato XML.



- En el área/ambiente global en RStudio verificamos que se creó el puntero `bm.doc` de clase 'XMLInternalDocument'.



Global Environment	
<b>Data</b>	
bm	List of 2
dat.1	12 obs. of 5 variables
dat.2	2 obs. of 6 variables
peru	List of 2
pob.es	Large list (159 elements, 1.9 Mb)
pob.esp	159 obs. of 6 variables
<b>values</b>	
bm.doc	External pointer of class 'XMLInternalDocument'
bm.url	"http://api.worldbank.org/v2/countries/all/ind..."
ine.url	"https://servicios.ine.es/wstempus/js/ES/DATOS..."

- Este puntero deberá recorrerse (pasar por cada un de los nodos que lo conforman). Para ello, debemos recuperar primero el nodo raíz a través de la instrucción `xmlRoot()`, pasándole el puntero.

```
root.node <- xmlRoot(bm.doc)
```

- Para verificar que Podemos recorrerlo, vemos el primer elemento o nodo del documento.

```
root.node[1]
```

Y obtenemos la siguiente salida en la consola (el primer elemento o nodo):

```
> root.node[1]
$data
<wb:data>
  <wb:indicator id="NY.GDP.MKTP.CD">GDP (current US$)</wb:indicator>
  <wb:country id="1A">Arab world</wb:country>
  <wb:countryiso3code>ARB</wb:countryiso3code>
  <wb:date>2020</wb:date>
  <wb:value/>
  <wb:unit/>
  <wb:obs_status/>
  <wb:decimal>0</wb:decimal>
</wb:data>

attr(,"class")
[1] "XMLInternalNodeList" "XMLNodeList"
> |
```

- Comparado con el formato CSV, aquí no obtenemos todos los datos del documento, sino que hay que acceder a ellos uno por uno.

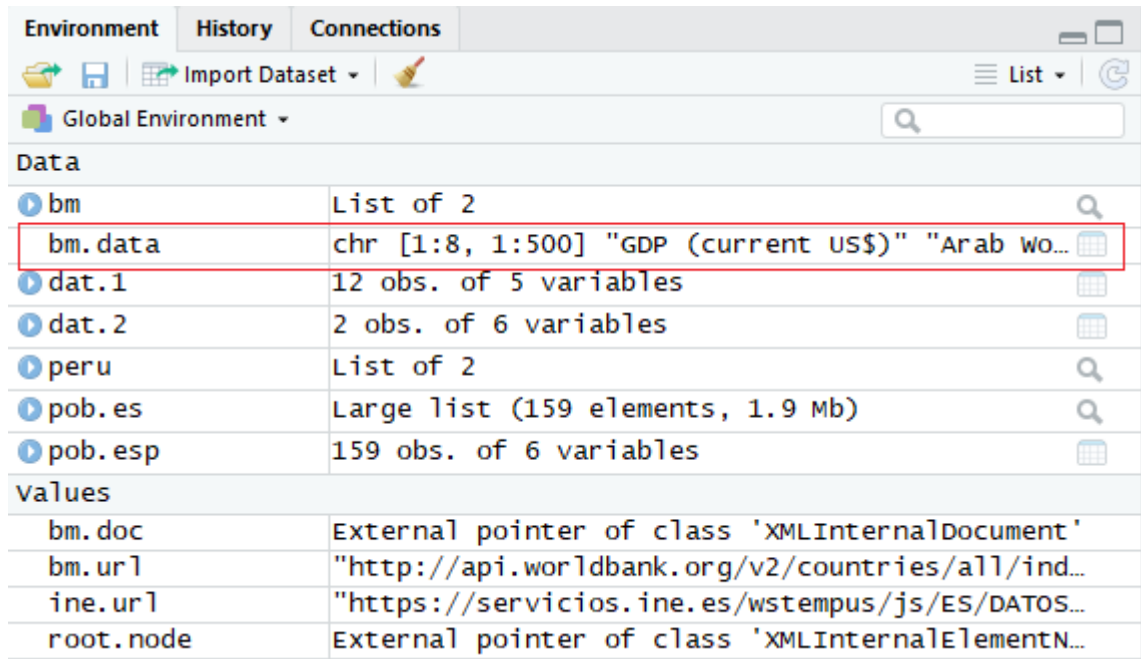
```
> root.node[2]
$data
<wb:data>
  <wb:indicator id="NY.GDP.MKTP.CD">GDP (current US$)</wb:indicator>
  <wb:country id="1A">Arab world</wb:country>
  <wb:countryiso3code>ARB</wb:countryiso3code>
  <wb:date>2019</wb:date>
  <wb:value>2817414584665.11</wb:value>
  <wb:unit/>
  <wb:obs_status/>
  <wb:decimal>0</wb:decimal>
</wb:data>

attr(,"class")
[1] "XMLInternalNodeList" "XMLNodeList"
> |
```

- Creamos un dataframe para acceder a todos los datos del xml, utilizando la instrucción **xmlSApply()**, y pasándole como argumento el nodo raíz e indicándole que le queremos aplicar una función a cada uno de sus elementos para extraer el valor. De esta forma:

```
bm.data <- xmlSApply(root.node, function(x) xmlSApply(x, xmlValue))
```

Vemos ahora creado el dataframe bm.data



Global Environment	
<b>Data</b>	
bm	List of 2
<b>bm.data</b>	chr [1:8, 1:500] "GDP (current US\$)" "Arab wo..."
dat.1	12 obs. of 5 variables
dat.2	2 obs. of 6 variables
peru	List of 2
pob.es	Large list (159 elements, 1.9 Mb)
pob.esp	159 obs. of 6 variables
<b>values</b>	
bm.doc	External pointer of class 'XMLInternalDocument'
bm.url	"http://api.worldbank.org/v2/countries/all/ind..."
ine.url	"https://servicios.ine.es/wstempus/js/ES/DATOS..."
root.node	External pointer of class 'XMLInternalElementN...

Y al consultarlo visualizamos lo siguiente:

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

01-analisis-json.R 01-analisis-json-files.R 01-analisis-xml.R bm.data pob.esp

	data	data	data	data	data	data	data	data
indicator	GDP (current US\$)	GDP (current US\$)	GDP (current US\$)	GDP (current US\$)	GDP (current US\$)	GDP (current US\$)	GDP (current US\$)	GDP (current US\$)
country	Arab World	Arab World	Caribbean small states	Caribbean small states	Central Europe and the Baltics	Central Europe and the Baltics	Early-demographic dividend	Early-demographic dividend
countryiso3code	ARB	ARB	CSS	CSS	CEB	CEB	EAR	EAR
date	2020	2019	2020	2019	2020	2019	2020	2019
value		2817414584665.11		77721714917.8506		1668851928931.89		11989867025219.9
unit								
obs_status								
decimal	0	0	0	0	0	0	0	0

- Observamos que los atributos se encuentran como filas y las observaciones como columnas. (este es el formato original como se recuperan los datos XML). Deberemos entonces transponer filas por columnas (las filas serán columnas y las columnas serán filas). Utilizaremos la instrucción `data.frame()` para crear un conjunto de datos y `t()` para transponer filas x columnas de `bm.data`. El parámetro `row.names = NULL` indica que las filas no tendrán nombre, y será reemplazado dicho valor por un numero correlativo.

```
bm.datos <- data.frame(t(bm.data), row.names = NULL)
```

- ¡Listo! Ahora si verificamos que nuestro dataset tienen 500 observaciones y 8 variables

Environment History Connections

Import Dataset

Global Environment

Data

bm	List of 2
bm.data	chr [1:8, 1:500] "GDP (current US\$)" "Arab wo..."
bm.datos	500 obs. of 8 variables
dat.1	12 obs. of 5 variables
dat.2	2 obs. of 6 variables
peru	List of 2
pob.es	Large list (159 elements, 1.9 Mb)
pob.esp	159 obs. of 6 variables
values	
bm.doc	External pointer of class 'XMLInternalDocument'
bm.url	"http://api.worldbank.org/v2/countries/all/ind..."
ine.url	"https://servicios.ine.es/wstempus/js/ES/DATOS..."
root.node	External pointer of class 'XMLInternalElementN...

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

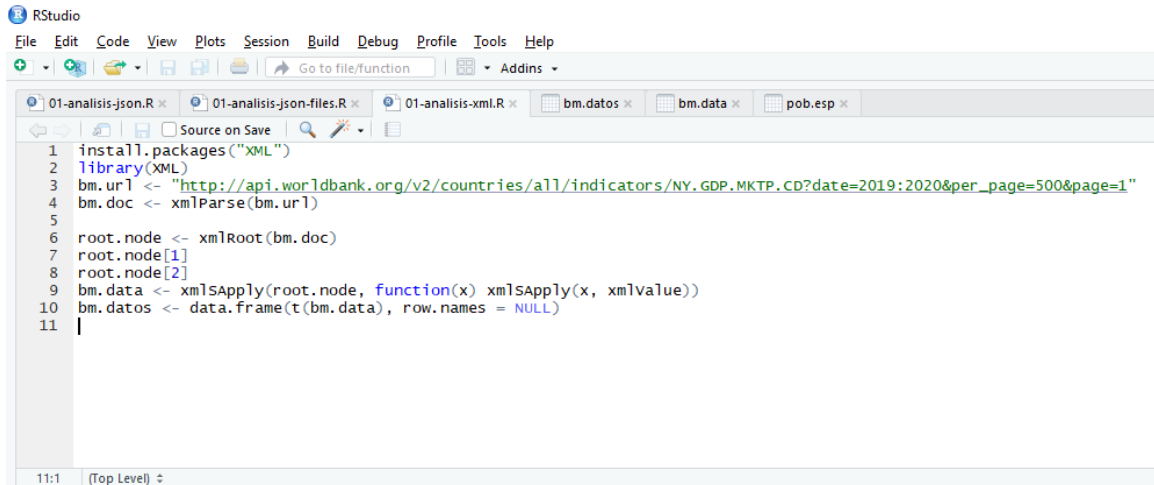
Go to file/function Addins

01-analisis-json.R 01-analisis-json-files.R 01-analisis-xml.R bm.datos bm.data pob.esp

	indicator	country	countryiso3code	date	value	unit	obs_status	decimal
399	GDP (current US\$)	Papua New Guinea	PNG	2019	24829107011.0701			0
400	GDP (current US\$)	Papua New Guinea	PNG	2019	24829107011.0701			0
401	GDP (current US\$)	Paraguay	PRY	2020				0
402	GDP (current US\$)	Paraguay	PRY	2019	38145288939.8488			0
403	GDP (current US\$)	Peru	PER	2020				0
404	GDP (current US\$)	Peru	PER	2019	226848050819.525			0
405	GDP (current US\$)	Philippines	PHL	2020				0
406	GDP (current US\$)	Philippines	PHL	2019	376795508678.853			0
407	GDP (current US\$)	Poland	POL	2020				0
408	GDP (current US\$)	Poland	POL	2019	595858207011.512			0
409	GDP (current US\$)	Portugal	PRT	2020				0

Showing 399 to 410 of 500 entries

El script en R debió haber quedado de esta forma:



```
1 install.packages("XML")
2 library(XML)
3 bm.url <- "http://api.worldbank.org/v2/countries/all/indicators/NY.GDP.MKTP.CD?date=2019:2020&per_page=500&page=1"
4 bm.doc <- xmlParse(bm.url)
5
6 root.node <- xmlRoot(bm.doc)
7 root.node[1]
8 root.node[2]
9 bm.data <- xmlSapply(root.node, function(x) xmlSapply(x, xmlValue))
10 bm.datos <- data.frame(t(bm.data), row.names = NULL)
11 |
```

## Ejercicio #2:

Realizar el mismo ejercicio, para:

(1) El archivo cd\_catalog.xml adjunto en el aula virtual.

Guardar el script con el nombre: 01-analisis-xml-file

## c) EXTRACCIÓN DE DATOS DESDE UN CSV

Un **CSV (COMMA-SEPARATED VALUES)** es un fichero de texto que almacena los datos en forma de columnas, separadas por coma y las filas se distinguen por saltos de línea. Este formato permite representar la información de forma sencilla.

**En R/RStudio ejecutaremos las siguientes instrucciones.**

- Creamos un archivo R en nuestro directorio de trabajo, al que llamaremos 01-analisis-csv
- La fuente de datos la obtenemos del archivo auto-mpg.csv ubicado en el aula virtual.
- Creamos la variable string autos y le asignamos el contenido del CSV utilizando la instrucción `read.csv()`

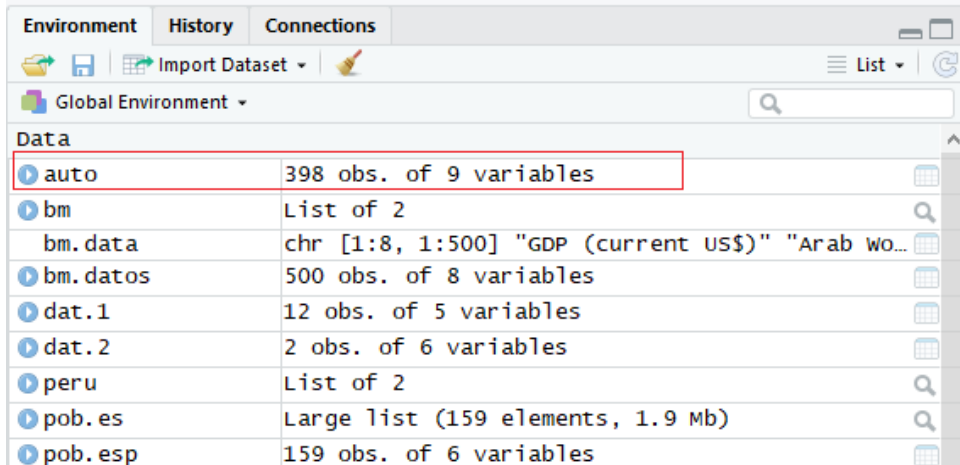
```
auto <- read.csv("data/auto-mpg.csv", header = TRUE, sep = ",")
```

Se considera la cabecera o título en cada columna y el separador en este caso en ",", pero en otro caso podría ser ";" o tabulador (`sep = "\t"`). Tanto `header = true` y `sep = ","` son valores por defecto, por tanto, se podrían omitir.

**Nota:** La instrucción `read.table()` utiliza el separador "espacio en blanco" para separar las diferentes columnas, por tanto, `read.csv()` es más especializado. Se recomienda utilizar como separador de columna el ";" dado que, en algunos casos, los valores numéricos utilizan la ","

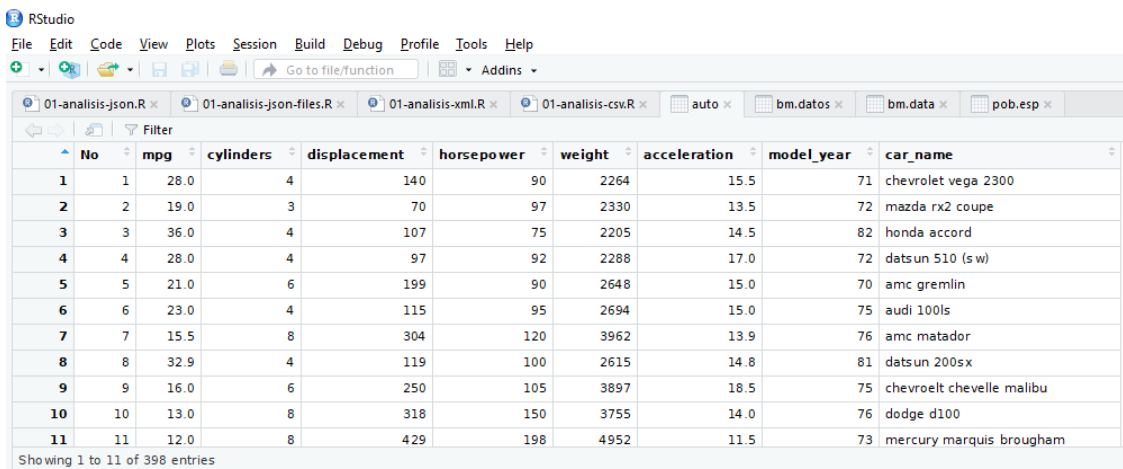
como separador decimal. La instrucción `read.csv2()` considera por defecto el separador “;” y decimal “,”.

Una vez ejecutada la instrucción, visualizamos que tenemos un dataframe con 398 observaciones y 9 variables.



Object	Description
auto	398 obs. of 9 variables
bm	List of 2
bm.data	chr [1:8, 1:500] "GDP (current US\$)" "Arab Wo..."
bm.datos	500 obs. of 8 variables
dat.1	12 obs. of 5 variables
dat.2	2 obs. of 6 variables
peru	List of 2
pob.es	Large list (159 elements, 1.9 Mb)
pob.esp	159 obs. of 6 variables

Cuyo contenido lo visualizamos con `view(auto)` o dando clic sobre el dataframe



	No	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	car_name
1	1	28.0	4	140	90	2264	15.5	71	chevrolet vega 2300
2	2	19.0	3	70	97	2330	13.5	72	mazda rx2 coupe
3	3	36.0	4	107	75	2205	14.5	82	honda accord
4	4	28.0	4	97	92	2288	17.0	72	datson 510 (s w)
5	5	21.0	6	199	90	2648	15.0	70	amc gremlin
6	6	23.0	4	115	95	2694	15.0	75	audi 100ls
7	7	15.5	8	304	120	3962	13.9	76	amc matador
8	8	32.9	4	119	100	2615	14.8	81	datson 200sx
9	9	16.0	6	250	105	3897	18.5	75	chevrolet chevelle malibu
10	10	13.0	8	318	150	3755	14.0	76	dodge d100
11	11	12.0	8	429	198	4952	11.5	73	mercury marquis brougham

Showing 1 to 11 of 398 entries

```
> names(auto)
[1] "No"           "mpg"          "cylinders"    "displacement" "horsepower"
[6] "weight"       "acceleration" "model_year"   "car_name"
```

#### d) LECTURA DE TABLAS INCRUSTADAS EN ARCHIVOS HTML

**HTML** (hypertext markup language) es un formato más especializado que XML que se utiliza para el desarrollo y creación de páginas web. Viene compuesto de una serie de etiquetas que el navegador interpreta.

**En R/RStudio ejecutaremos las siguientes instrucciones.**

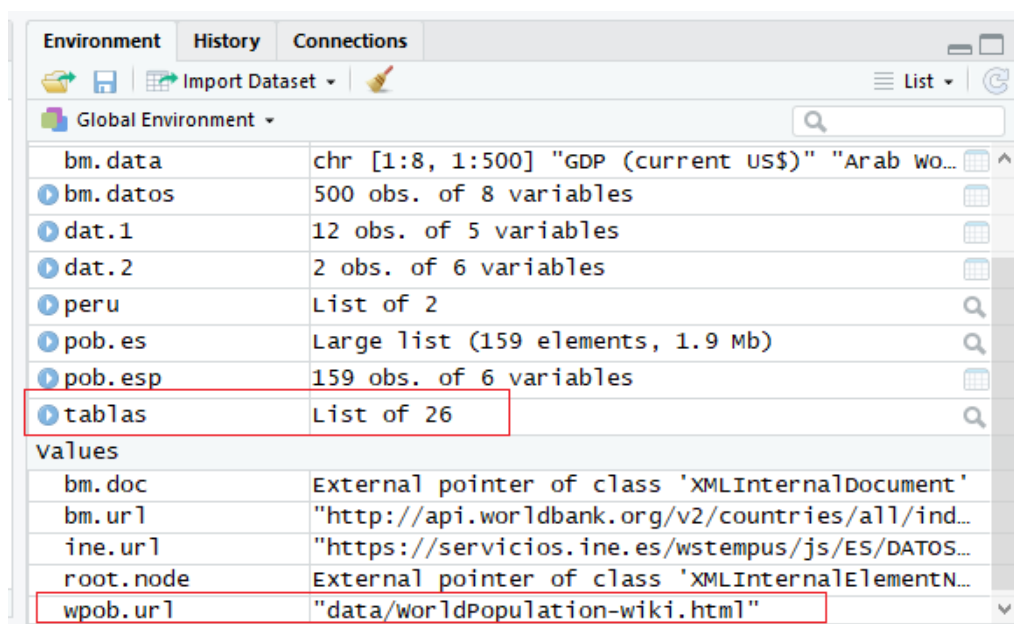
- Creamos un archivo R en nuestro directorio de trabajo, al que llamaremos 01-analisis-html
- La fuente de datos la obtenemos del archivo **WorldPopulation-wiki.html** ubicado en el aula virtual.
- Creamos la variable string wpob.url y le asignamos el archivo html

```
wpob.url <- "data/WorldPopulation-wiki.html"
```

- Extraemos todas las tablas que contiene este html con la instrucción **readHTMLTable()**

```
tablas <- readHTMLTable(wpob.url)
```

- Visualizamos lo obtenido en el ambiente global:



- Como resultado, dentro de este html existen 26 tablas con distinto contenido:

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

01-analisis-json.R x 01-analisis-json-files.R x 01-analisis-xml.R x 01-analisis-csv.R x 01-analisis-html.R\* x tablas x

Show Attributes

Name	Type	Value
Date	factor	Factor with 10 levels: "September 9, 2017", "September 9, 2017", "September 9, 2, ...
Approx. % of world ...	factor	Factor with 10 levels: "18.3%", "17.5%", "4.31%", "3.46%", "2.76%", "2.75%", ...
Source	factor	Factor with 10 levels: "[91]", "[92]", "[93]", "[94]", "[95]", "[96]", ...
10 most densely popul...	list [10 x 5] (S3: data.frame)	A data.frame with 10 rows and 5 columns
Rank	factor	Factor with 10 levels: "1", "2", "3", "4", "5", "6", ...
Country	factor	Factor with 10 levels: "Singapore", "Bangladesh", "Taiwan", "South Korea", "Leba, ...
Population	factor	Factor with 10 levels: "5,607,300", "163,110,000", "23,554,803", "51,446,201", " , ...
Area (km2)	factor	Factor with 10 levels: "710", "143,998", "36,190", "99,538", "10,452", "26,338", ...
Density (Pop. per km2)	factor	Factor with 10 levels: "7,898", "1,133", "651", "517", "582", "439", ...
Countries ranking high...	list [10 x 6] (S3: data.frame)	A data.frame with 10 rows and 6 columns
NULL	NULL	Pairlist of length 0

tablas

- Elegiremos la quinta tabla que contienen la relación de los 10 países más poblados del mundo. Como la variable tablas es una lista de listas, accederemos al índice 6 dentro de doble corchete:

```
most_wpob <- tablas[[5]]
```

```
head(most_wpob,3)
```

```
> head(most_wpob,3)
  Rank Country Population % of world Date Source(official or UN)\n
1    1   China 1,407,591,320 17.9% 20 Apr 2021 National population clock[89]
2    2   India 1,375,956,205 17.5% 20 Apr 2021 National population clock[90]
3    3 United States 331,532,247 4.22% 20 Apr 2021 National population clock[91]
```

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

01-analisis-json.R x 01-analisis-json-files.R x 01-analisis-xml.R x 01-analisis-csv.R x 01-an...

Filter

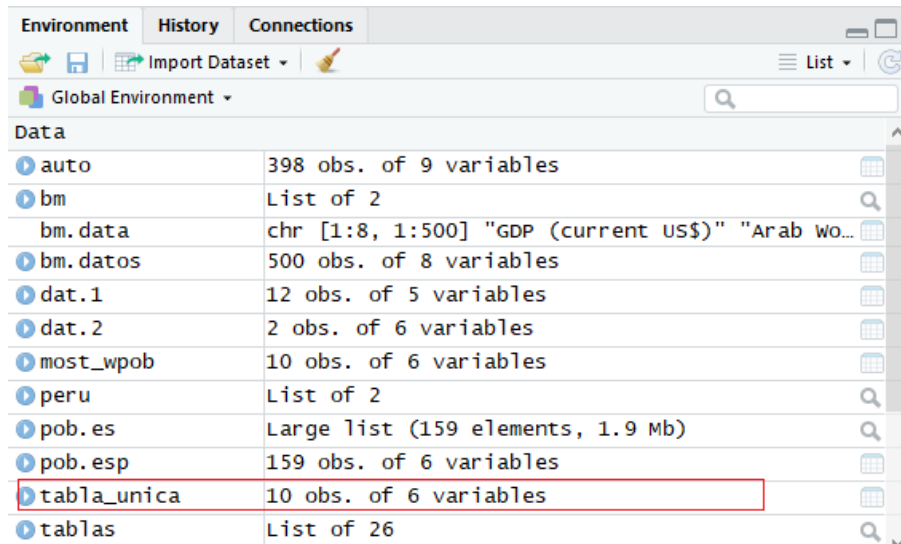
	Rank	Country	Population	% of world	Date	Source(official or UN)
1	1	China	1,407,591,320	17.9%	20 Apr 2021	National population clock[89]
2	2	India	1,375,956,205	17.5%	20 Apr 2021	National population clock[90]
3	3	United States	331,532,247	4.22%	20 Apr 2021	National population clock[91]
4	4	Indonesia	269,603,400	3.43%	1 Jul 2020	National annual projection[92]
5	5	Pakistan	220,892,331	2.81%	1 Jul 2020	UN Projection[93]
6	6	Brazil	213,026,892	2.71%	20 Apr 2021	National population clock[94]
7	7	Nigeria	206,139,587	2.62%	1 Jul 2020	UN Projection[93]
8	8	Bangladesh	170,526,678	2.17%	20 Apr 2021	National population clock[95]
9	9	Russia	146,748,590	1.87%	1 Jan 2020	National annual estimate[96]
10	10	Mexico	127,792,286	1.63%	1 Jul 2020	National annual projection[97]

Showing 1 to 10 of 10 entries

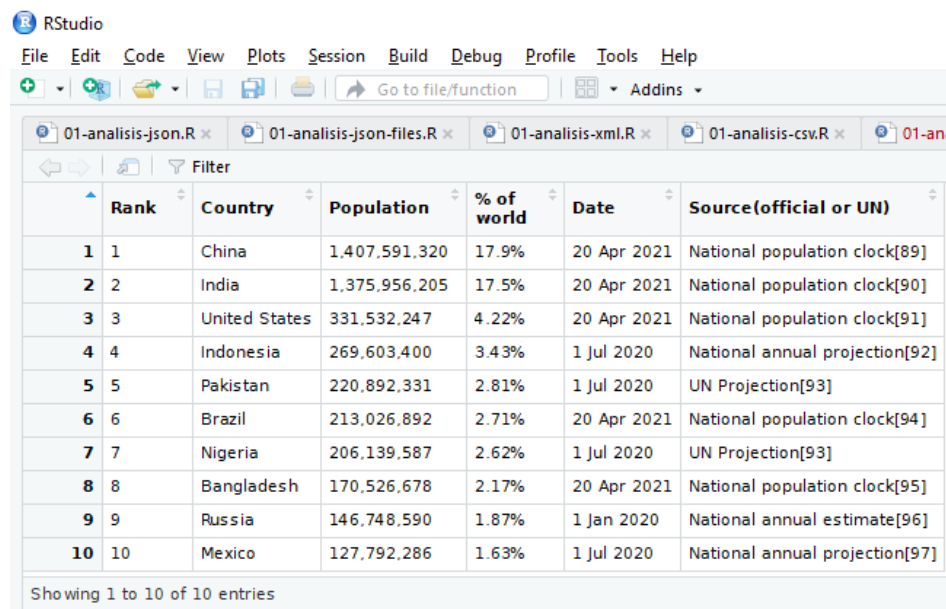
- Si no queremos cargar todas las tablas de una página web y conocemos el índice de su ubicación dentro de la página, utilizamos el parámetro `which`. Este acceso es más rápido si la página fuera muy pesada y tienen muchas tablas.

```
tabla_unica <- readHTMLTable(wpob.url, which = 5)
```

- Cuyo resultado da la siguiente visualización:



Object	Description
auto	398 obs. of 9 variables
bm	List of 2
bm.data	chr [1:8, 1:500] "GDP (current US\$)" "Arab wo..."
bm.datos	500 obs. of 8 variables
dat.1	12 obs. of 5 variables
dat.2	2 obs. of 6 variables
most_wpob	10 obs. of 6 variables
peru	List of 2
pob.es	Large list (159 elements, 1.9 Mb)
pob.esp	159 obs. of 6 variables
tabla_unica	10 obs. of 6 variables
tablas	List of 26

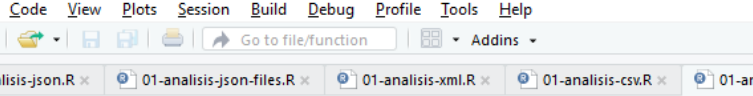


	Rank	Country	Population	% of world	Date	Source (official or UN)
1	1	China	1,407,591,320	17.9%	20 Apr 2021	National population clock[89]
2	2	India	1,375,956,205	17.5%	20 Apr 2021	National population clock[90]
3	3	United States	331,532,247	4.22%	20 Apr 2021	National population clock[91]
4	4	Indonesia	269,603,400	3.43%	1 Jul 2020	National annual projection[92]
5	5	Pakistan	220,892,331	2.81%	1 Jul 2020	UN Projection[93]
6	6	Brazil	213,026,892	2.71%	20 Apr 2021	National population clock[94]
7	7	Nigeria	206,139,587	2.62%	1 Jul 2020	UN Projection[93]
8	8	Bangladesh	170,526,678	2.17%	20 Apr 2021	National population clock[95]
9	9	Russia	146,748,590	1.87%	1 Jan 2020	National annual estimate[96]
10	10	Mexico	127,792,286	1.63%	1 Jul 2020	National annual projection[97]

Showing 1 to 10 of 10 entries



- Código en R:



The screenshot shows the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar labeled 'Go to file/function'. The tab bar shows several open files: '01-analysis-json.R', '01-analysis-json-files.R', '01-analysis-xml.R', '01-analysis-csv.R', and '01-analysis-html.R'. The active file is '01-analysis-html.R'. The script editor displays the following R code:

```
1 wpob.url <- "data/worldPopulation-wiki.html"
2
3 tablas <- readHTMLTable(wpob.url)
4
5 most_wpob <- tablas[[5]]
6
7 head(most_wpob, 3)
8
9 tabla_unica <- readHTMLTable(wpob.url, which = 5)
10 _
```