
Caso práctico: canales de YouTube

PID_00271957

Àngel Ollé Blázquez

Tiempo mínimo de dedicación recomendado: 1 hora



**Àngel Ollé Blázquez**

Ingeniero técnico en Informática de gestión por la Universidad Rovira i Virgili (URV). Ingeniero en Informática y máster en Software libre por la Universitat Oberta de Catalunya (UOC). Actualmente trabaja como ingeniero de software y participa en proyectos *open source*. Anteriormente desarrolló su actividad profesional en el sector de servicios y consultoría IT por EMEA.

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por el profesor: Julià Minguillón Alfonso (2020)

Primera edición: febrero 2020
© Àngel Ollé Blázquez
Todos los derechos reservados
© de esta edición, FUOC, 2020
Avda. Tibidabo, 39-43, 08035 Barcelona
Realización editorial: FUOC

Ninguna parte de esta publicación, incluido el diseño general y la cubierta, puede ser copiada, reproducida, almacenada o transmitida de ninguna forma, ni por ningún medio, sea este eléctrico, químico, mecánico, óptico, grabación, fotocopia, o cualquier otro, sin la previa autorización escrita de los titulares de los derechos.

Índice

1. Introducción.....	5
2. Inicio del caso práctico.....	6

1. Introducción

En este caso práctico se profundiza en los medios de obtención, transformación y producción de los datos. Además, ejemplificamos las diferentes alternativas que se tienen a la hora de extraer e interpretar los resultados de los datos.

En este caso, los datos se adquieren a partir de consultas a la API de YouTube. La interacción con esta API es más compleja y muestra cómo solucionar los problemas y particularidades mediante *scripts* con Bash. Además, se muestra cómo fusionar los ficheros y cribrar los datos: por un lado, con la introducción de las herramientas proporcionadas con CSVKit y, por otro, con Grep y el uso de Sed o AWK para la normalización y sustituciones. También se introduce el uso de un *parser* de XML por línea de comandos y las expresiones *XPath*.

Finalmente, se hacen representaciones gráficas de las conclusiones en las que, transversalmente, se expone cómo funciona la interacción y la ejecución de *scripts* en otros lenguajes desde Bash.

2. Inicio del caso práctico

Fuente de datos	YouTube vía Google API: https://developers.google.com/youtube/v3/
Requisitos previos	<ul style="list-style-type: none"> • Cuenta de Google: https://www.google.com/accounts/NewAccount • <i>Token</i> de acceso

Obtener un *token* de acceso para YouTube

1) Visitamos: <https://code.google.com/apis/console>

2) Creamos un proyecto:

Nombre de proyecto *

ID del proyecto: uoc1-253021.No se puede cambiar más adelante. [EDITAR](#)

3) Creamos las credenciales:

<https://console.developers.google.com/apis/library/youtube.googleapis.com>.

Clicamos en el botón «Habilitar» y después en «Crear credenciales».

Información general INHABILITAR API

Es posible que necesites credenciales para usar esta API. Haz clic en [Crear credenciales](#) para empezar. [CREAR CREDENCIALES](#)

Detalles

Nombre
YouTube Data API v3

De
Google

Nombre del servicio
youtube.googleapis.com

Información general
The YouTube Data API v3 is an API that provides access to YouTube data, such as videos, playlists, and channels.

Estado de activación
Habilitada

Tráfico por código de respuesta

Petición/s (promedio de 2 h)

No hay datos disponibles del periodo.

Seguidamente, creamos unas credenciales para una entidad no UI, ya que usaremos los comandos del sistema operativo (curl, bash, etc.).

Credenciales

Añadir credenciales al proyecto

1 Averigua qué tipo de credenciales necesitas

Te ayudaremos a configurar las credenciales adecuadas.

Puedes saltarte este paso y crear una [clave de API](#), un [ID de cliente](#) o una [cuenta de servicio](#).

¿Qué API estás utilizando?

Las API utilizan diversas plataformas de autorización, y se pueden restringir algunas credenciales para que solo llamen a ciertas API.

YouTube Data API v3

¿Desde dónde llamarás a la API?

Para restringir las credenciales, se puede tener en cuenta el contexto de la llamada. No obstante, no es seguro usar algunas credenciales en contextos determinados.

Otra entidad que no sea UI (por ejemplo, tarea cron, ...)

¿A qué tipo de datos accederás?

En función del tipo de datos que solicites, se precisan unas credenciales determinadas para autorizar el acceso.

Datos públicos

Accede a datos que facilita la API y están disponibles públicamente

Datos de usuario

Accede a datos pertenecientes a un usuario de Google (con su permiso)

[¿Qué credenciales necesito?](#)

2 Obtener credenciales

Cancelar

Credenciales

Añadir credenciales al proyecto


Averigua qué tipo de credenciales necesitas

Llamar a YouTube Data API v3 desde una plataforma sin UI

2 Obtener credenciales

Esta es tu clave de API

<...>

 Te recomendamos restringir esta clave antes de utilizarla en producción. Las restricciones limitan los sitios web, direcciones IP o aplicaciones que pueden llamar a las API con esta clave.

[Restringir la clave](#)

Listo

Cancelar

Ya tenemos el *token* de acceso: <https://console.developers.google.com/apis/credentials>

Copiamos el *access token* y hacemos una petición `curl` para comprobar que todo funciona correctamente.

La petición la haremos en el canal de los *mossos d'esquadra*, la policía autonómica catalana, y la consulta será la obtención de los datos del canal:

```
curl \
  `https://www.googleapis.com/youtube/v3/channels?part=snippet%2CcontentDetails%2Cstatistics&id=UC_x5XG1OV2P6uZZ5FSM9Ttw&key=[API_KEY]' \
  --header 'Accept: application/json' \
  --compressed
```

Canal de los mossos:

<https://www.youtube.com/channel/UC5cri6CvVLNVmF2C3GbgJTw>

Resultado:

```
{
  "kind": "youtube#channelListResponse",
  "pageInfo": {
    "totalResults": 1,
    "resultsPerPage": 1
  },
  "items": [
    {
      "kind": "youtube#channel",
      "id": "UC5cri6CvVLNVmF2C3GbgJTw",
      "snippet": {
        "title": "Mossos",
        "description": "Benvinguts al canal YouTube del cos de Mossos d'Esquadra. Policia de Catalunya a les xarxes socials. Trobareu consells de seguretat, operatius policials i coneixereu el cos policial i les diferents unitats que som al servei de la ciutadania.",
        "customUrl": "mossoscatalunya",
        "publishedAt": "2010-02-03T09:23:44.000Z",
        "thumbnails": {
          "default": {
            "url": "https://yt3.ggpht.com/a/AGF-178wL9w4xW7nIscqpiI55Pes17xfSv3KbLkE5Q=s88-c-k-c0xffffffff-no-rj-mo",
            "width": 88,
            "height": 88
          },
          "medium": {
            "url": "https://yt3.ggpht.com/a/AGF-178wL9w4xW7nIscqpiI55Pes17xfSv3KbLkE5Q=s240-c-k-c0xffffffff-no-rj-mo",
            "width": 240,
            "height": 240
          },
          "high": {
            "url": "https://yt3.ggpht.com/a/AGF-178wL9w4xW7nIscqpiI55Pes17xfSv3KbLkE5Q=s800-c-k-c0xffffffff-no-rj-mo",
            "width": 800,
            "height": 800
          }
        },
        "localized": {
          "title": "Mossos",
          "description": "Benvinguts al canal YouTube del cos de Mossos d'Esquadra. Policia de Catalunya a les xarxes socials. Trobareu consells de seguretat, operatius policials i coneixereu el cos policial i les diferents unitats que som al servei de la ciutadania."
        }
      }
    }
  ]
}
```



```
},
"contentDetails": {
  "relatedPlaylists": {
    "uploads": "UU5cri6CvVLNVmF2C3GbgJTw",
    "watchHistory": "HL",
    "watchLater": "WL"
  }
},
"statistics": {
  "viewCount": "553058",
  "commentCount": "0",
  "subscriberCount": "1787",
  "hiddenSubscriberCount": false,
  "videoCount": "241"
}
}
]
```

Podemos ver que la API nos ha devuelto los datos básicos del canal. Con este resultado podemos confirmar que la configuración de Google funciona correctamente y ya podemos utilizar la API para extraer los datos.

Ahora obtendremos el *data set* de la lista de *snippets* con la información asociada. La API soporta un máximo de cincuenta resultados por consulta, por lo que tendremos que hacer la captura de datos empleando la paginación: <https://developers.google.com/youtube/v3/docs/search/list>.

`pagetoken` es el parámetro que identifica la página.

La consulta tendrá la forma inicial siguiente:

```
curl \
'https://www.googleapis.com/youtube/v3/search?part=snippet&channelId=
UC5cri6CvVLNVmF2C3GbgJTw&maxResults=50&key=[API_KEY]' \
--header 'Accept: application/json' \
--compressed
```

Y la respuesta:

```
{
  "kind": "youtube#searchListResponse",
  "nextPageToken": "CDIQAA",
  "regionCode": "ES",
  "pageInfo": {
    "totalResults": 252,
    "resultsPerPage": 50
  },
  "items": [
    {
<...>
  ]
}
```

La lista de ítems serán los cincuenta vídeos de la página. Pero queremos todos los datos, por lo que tendremos que construir un pequeño *script* que haga las consultas y vaya cambiando de página hasta que no queden más datos que obtener.

Un posible *script* puede ser el siguiente:

```
#!/bin/bash

canal=UC5scri6CvVLNVmF2C3GbgJTw
resultats=50
key=<API_KEY>
res=resultat.json

fcurl() {
  curl \
    "https://www.googleapis.com/youtube/v3/search?part=snippet&channelId=$canal&maxResults=$resultats&pageToken=$pagetoken&key=$key" \
    --header 'Accept: application/json' \
    --compressed \
    -o "$res.$i" 2>/dev/null
}

i=0
while ;; do
  pagetoken=$(jq -r '.nextPageToken' "$res.$i" 2>/dev/null)
  [[ ${pagetoken} == null ]] && break
  (( i++ ))
  fcurl
done
```

El proceso nos dejará varios ficheros json con formato `resultat.json.<#numero>`

```
resultat.json.1
resultat.json.2
resultat.json.3
resultat.json.4
resultat.json.5
```

Para todos los ficheros, extraemos la fecha de publicación, el título y la descripción de cada vídeo. Creamos un csv resultante:

```
echo "Data,Titol,Descripcio" > mossos.csv

jq -r '.items[].snippet|"\(.publishedAt)", "\(.title)", "\(.description)"'
resultat.json.* >> mossos.csv
```

Normalizaremos la fecha en formato final «día-mes-año» y la hora la descartaremos porque no es relevante para este caso práctico.

```
sed -i 's/"\([0-9]\{4\}\)-\([0-9]\{2\}\)-\([0-9]\{2\}\)T\(.*)Z"/"\3-\2-\1",\5/g'
mossos.csv
```

Ahora instalaremos **CSVKit**, que es un conjunto de herramientas para trabajar con CSV y está disponible en los repositorios de Debian y Ubuntu:

```
apt install csvkit
```

CSVKit proporciona el comando `csvstat`, que muestra las estadísticas más descriptivas del fichero csv. Lo procesaremos previamente con `csvcut` para evitar errores de formato, puesto que en algunas líneas de la descripción hay dobles comillas no situadas correctamente o con el carácter erróneo.

```
csvcut mossos.csv | csvstat > stat.txt
```

El fichero «stat.txt» contendrá unas estadísticas básicas sobre cada columna del csv:

```
1. "Data"

Type of data:      Text
Contains null values: False
Unique values:    151
Longest value:    10 characters
Most common values: 23-08-2016 (8x)
                  10-03-2017 (7x)
                  09-08-2017 (7x)
                  10-07-2018 (6x)
                  28-02-2017 (6x)

2. "Titol"

Type of data:      Text
Contains null values: False
Unique values:    226
Longest value:    109 characters

<...>

3. "Descripcio"

Type of data:      Text
Contains null values: False
Unique values:    195
Longest value:    171 characters

<...>

Row count: 239
```

Con esta estadística podemos ver que el día que se publicaron más vídeos fue el 23-08-2016, con ocho vídeos publicados.

En este canal se hace poco uso de los *hashtags*. Para hacernos una idea, los podemos extraer con la ayuda de `sed` y `egrep`, y el *hashtag* que más predomina es el de `#tufascampanya`:

```
$ cat mossos.csv | sed 's/ /\n/g' | egrep "^#[a-zA-Z0-9]+" | tr -d ",.\"" | sort | uniq -c | sort -n -r

16 #tufascampanya
 3 #VoltaCatalunya
 2 #seguetat
 2 #SecurityForum
 2 #Mossos
 1 #violenciadeGenere
 1 #Video
 1 #video
 1 #TEDAX-NRBQ
 1 #sommossos
 1 #Premia
 1 #Muntanya
 1 #MesMossos
 1 #Masnou
 1 #LlistaNegra
 1 #Israel
 1 #GRÀCIES
 1 #Facebook
 1 #Esquadres
 1 #DiaDones2019
 1 #cosadetots!
 1 #control
 1 #ComComençaTot
```

```
1 #Android
```

Ahora, haremos dos representaciones visuales de los datos: la primera con la cantidad de producciones de vídeos por año y la segunda con la cantidad de producciones relacionadas con las poblaciones de Barcelona.

Los gráficos los haremos llamando un *script* en *R* desde Bash. Para instalar *R* y *ggplot2*, el paquete que usa el *script* para construir los gráficos, lo haremos con:

```
apt install r-base r-cran-ggplot2
```

En primer lugar, usaremos `csvcut` para arreglar los problemas de formato (comillas dobles mal situadas) que tiene el fichero, al igual que antes, pero esta vez guardaremos los resultados en un fichero para poder trabajar de manera más cómoda.

```
$ csvcut mossos.csv > mossos_format.csv
```

Ahora crearemos el esquema automáticamente con `csvsql` para poder consultar los datos por SQL.

```
$ csvsql mossos_format.csv

CREATE TABLE mossos_format (
  "Data" VARCHAR(10) NOT NULL,
  "Titol" VARCHAR(109) NOT NULL,
  "Descripcio" VARCHAR(171) NOT NULL
);
```

Generamos la consulta SQL para extraer los años y la cantidad de vídeos por año:

```
$ csvsql --query "select substr(Data,7,7) as Any, count(substr(Data,7,7))
as Total from mossos_format group by substr(Data,7,7)" mossos_format.csv

Any,Total
2011,5
2012,7
2013,18
2014,10
2015,7
2016,26
2017,62
2018,81
2019,23
```

Guardamos los resultados en un fichero.

```
$ csvsql --query "select substr(Data,7,7) as Any, count(substr(Data,7,7))
as Total from mossos_format group by substr(Data,7,7)" mossos_format.csv > mossos_videos_any.csv
```

El *script* se encarga de hacer el gráfico a partir de los argumentos de entrada:

- fichero de datos en csv,
- coordenada X,
- coordenada Y,
- escala del eje X,
- escala del eje Y y

- fichero de salida.

```
#!/usr/bin/env Rscript
library(ggplot2)

args = commandArgs(trailingOnly = TRUE)

if (length(args) != 6) {
  stop("Calen 6 arguments: fitxer de dades, coordenada X, coordenada Y, primer valor desitjat
  escalat eix X, darrer valor desitjat d'escalat eix X, fitxer de sortida (png)")
}

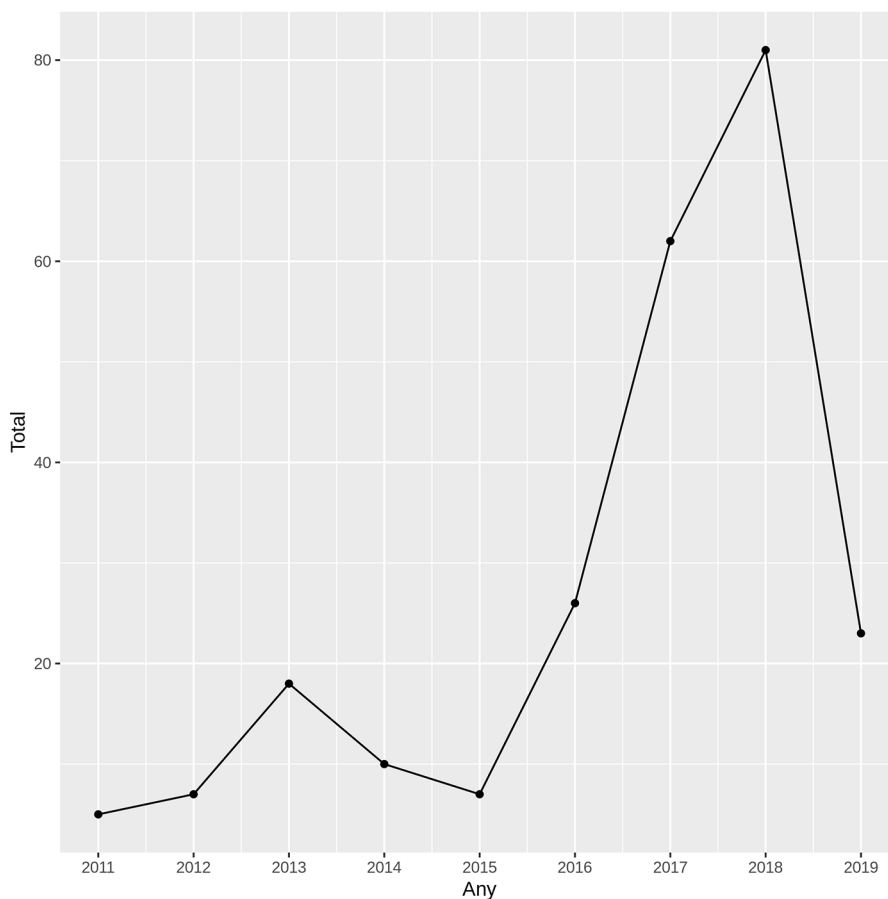
d <- read.csv(args[1])
p <- ggplot(data = d, aes_(x = as.name(args[2]), y = as.name(args[3]))) + geom_point() +
  geom_line() + scale_x_continuous(breaks = seq(args[4], args[5]))

ggsave(args[6], plot = p)
```

Lo ejecutaremos pasándole los argumentos desde Bash:

```
$ chmod +x linies.R
$ ./linies.R mossos_videos_any.csv Any Total 2011 2019 resultat.png
```

El *script* creará un fichero png de salida con el gráfico:



Veamos ahora cuántos vídeos hay relacionados con las poblaciones de Barcelona. Para ello, cruzaremos los datos con los de la Wikipedia. En este escenario, en lugar de dirigirnos hacia la API de la Wikipedia, obtendremos sus datos en html y utilizaremos `xmllint`, presente en la mayoría de distribuciones, para hacer el *parsing* de los datos que nos interesan.

La url es:

```
https://es.wikipedia.org/
wiki/Anexo:Municipios_de_la_provincia_de_Barcelona
```

Obtendremos las poblaciones de la segunda columna, así que podemos usar la expresión XPath siguiente:

```
/html/body/div[3]/div[3]/div[4]/div/table/tbody/tr/td[2]
```

Con `sed` se extrae del resultado el nombre de la población, eliminando las etiquetas que no nos interesan.

El comando final será:

```
$ curl https://es.wikipedia.org/wiki/Anexo:Municipios_de_la_provincia_de_Barcelona 2>/dev/null |\
xmllint --html --xpath '/html/body/div[3]/div[3]/div[4]/div/table/tbody/tr/td[2]' - |\
sed 's/<.*>\(.*\)\/\1/' \
> poblacions-bcn.txt
```

Para evitar problemas con la codificación durante el filtraje, eliminaremos la acentuación:

```
$ sed -i 'y/àèòéíóúï/aeoeiouï/' poblacions-bcn.txt
$ sed -i 'y/àèòéíóúï/aeoeiouï/' mossos.csv
```

Ahora cruzaremos los datos de las poblaciones y guardaremos el resultado en un fichero. Escogemos añadir una columna más al csv, con el campo población al principio de cada fila. También se podría resolver creando solo un fichero al lado, con una única columna que tenga el resultado secuencial de cada población.

Conviene tener en cuenta que una línea puede contener ninguna, una o más poblaciones.

```
$ while read -r poblacio; do grep "$poblacio" mossos.csv | sed "s/^/\"$poblacio\" , /" ;
done < poblacions-bcn.txt > filtre_poblacions.csv
```

Quedará:

```
"Arenys de Mar", "05-02-2018", "Unitat Aquatica retira una xarxa de pesca a la deriva",
"Es retira una xarxa de pesca perduda al fons mari d'Arenys de Mar que constituïa un perill
per a les persones i l'ecosistema. L'accio, coordinada entre la Unitat ..."
<...>
```

Para ilustrar otra manera diferente de componer el fichero resultante, en este ejemplo lo haremos con las herramientas de la *shell* y Bash. Pero también se podría hacer cargando el resultado con `csvsql` y realizando la consulta SQL:

```
select Poblacio, count(*) as Total from filtre_poblacions_format group by Poblacio)
```

Se extraen las columnas y se cuentan las ocurrencias con `sed`:

```
$ echo "Total,Poblacio" > mossos_poblacions.csv
$ csvcut -c 1 filtre_poblacions.csv | sort | uniq -c | sed 's/^[ ]\+//; s/\ /,/' >>
mossos_poblacions.csv
```

O también podemos utilizar `awk`:

```
$ echo "Total,Poblacio" > mossos_poblacions.csv
$ csvcut -c 1 filtre_poblacions.csv | sort | uniq -c | awk '{ gsub (/^[ ]\+/, "", $0); sub
(/ /, ",", $0); print}' >> mossos_poblacions.csv
```

El fichero resultante queda como sigue:

```
$ cat mossos_poblacions.csv

Total,Poblacio
2,Arenys de Mar
28,Barcelona
1,Manresa
1,Martorell
2,Montcada i Reixac
1,Sabadell
1,Sallent
1,Seva
1,Terrassa
1,Vilassar de Mar
```

El *script* se encarga de hacer el gráfico a partir de los argumentos de entrada:

- fichero de datos en csv,
- coordenada X,
- coordenada Y,
- cumplimentación y
- fichero de salida.

```
#!/usr/bin/env Rscript

library(ggplot2)

args = commandArgs(trailingOnly = TRUE)

if (length(args) != 5) {
  stop("Calen 5 arguments: fitxer, coordenada X, coordenada Y, emplenament i fitxer de sortida")
}

d <- read.csv(args[1])
p <- ggplot(data = d, aes_(x = as.name(args[2]), y = as.name(args[3]), fill = as.name(args[4])))
+ geom_bar(stat = 'identity') + theme(axis.text.x = element_blank(), axis.ticks.x =
element_blank())

ggsave(args[5], plot=p)
```

Ejecutamos:

```
$ chmod +x ./barres.R  
$ ./barres.R mossos_poblacions.csv Poblacio Total Poblacio resultat2.png
```

El *script* creará un fichero png de salida con el gráfico:

