Cas pràctic: canals de YouTube

PID_00271955

Àngel Ollé Blázquez

Temps mínim de dedicació recomanat: 1 hora

Universitat Oberta de Catalunya

Àngel Ollé Blázquez

Enginyer tècnic en Informàtica de gestió per la Universitat Rovira i Virgili (URV). Enginyer en Informàtica i màster en Programari Lliure per la Universitat Oberta de Catalunya (UOC). Actualment treballa com a enginyer de programari i participa en projectes *open source*. Anteriorment ha desenvolupat la seva activitat professional en el sector de serveis i consultoria IT per EMEA.

L'encàrrec i la creació d'aquest recurs d'aprenentatge UOC han estat coordinats pel professor: Julià Minguillón Alfonso (2020)

Primera edició: febrer 2020

Av. Tibidabo, 39-43, 08035 Barcelona
Realització editorial: FUOC

Índex

1. Introducció

En aquest cas pràctic s'aprofundeix en els mitjans d'obtenció, transformació i producció de les dades. A més a més, exemplifiquem les diferents alternatives que es tenen a l'hora d'extreure i interpretar els resultats de les dades.

En aquest cas les dades s'adquireixen a partir de consultes a l'API de YouTube. La interacció amb aquesta API és més complexa i mostra com solucionar els problemes i particularitats mitjançant *scripts* amb Bash. Tanmateix, es mostra com fusionar els fitxers i cribrar les dades: per una banda, amb la introducció de les eines proporcionades amb CSVKit i, per altra banda, amb Grep i l'ús de Sed o AWK per a la normalització i substitucions. També s'introdueix l'ús d'un *parser* d'XML per línia de comandes i les expressions *XPath*.

Finalment, es fan representacions gràfiques de les conclusions en què transversalment s'exposa com funciona la interacció i execució d'*scripts* en altres llenguatges des de Bash.

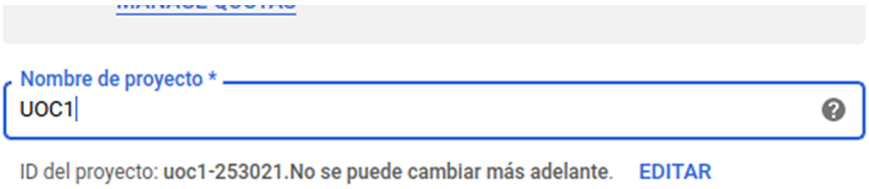
2. Inici del cas pràctic

Font de dades	YouTube via la Google API: https://developers.google.com/youtube/v3/
Requisits previs	• Compte de Google: https://www.google.com/accounts/NewAccount • *Token* d'accés

Obtenir un *token* d'accés per a YouTube

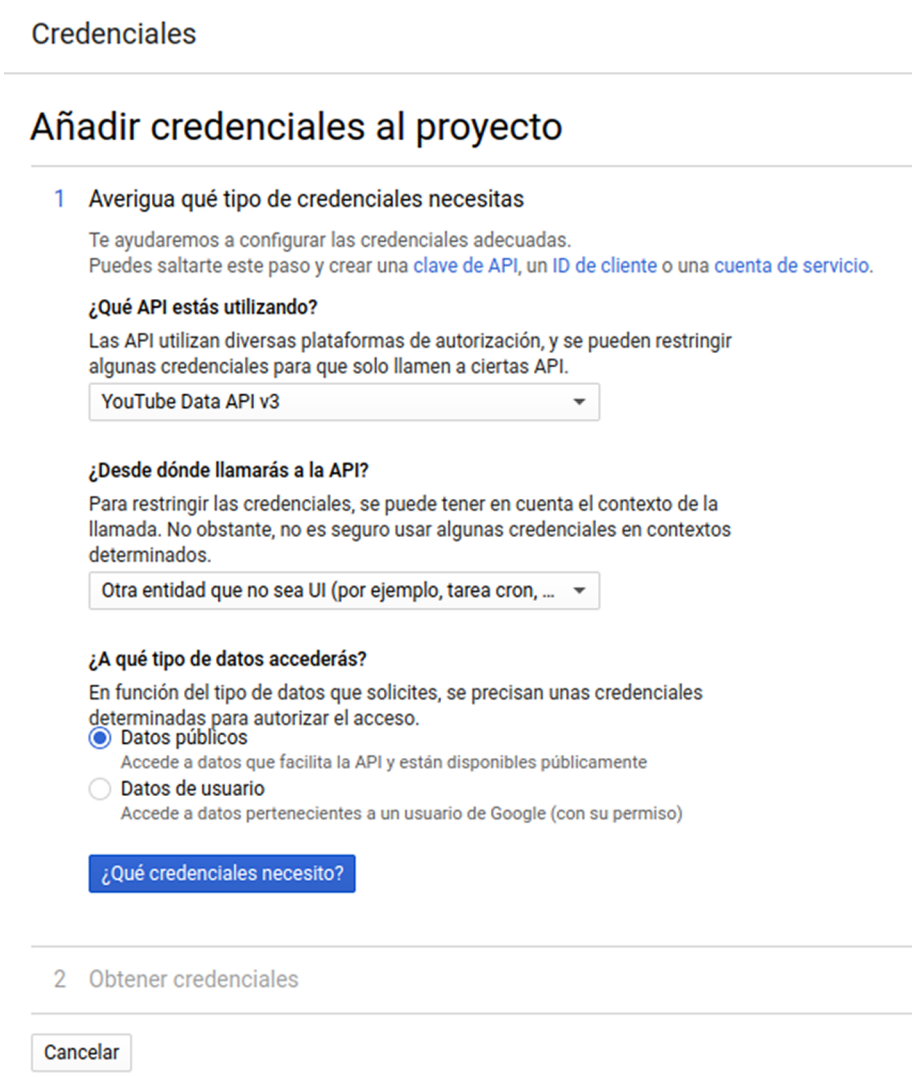
1) Visitem: https://code.google.com/apis/console

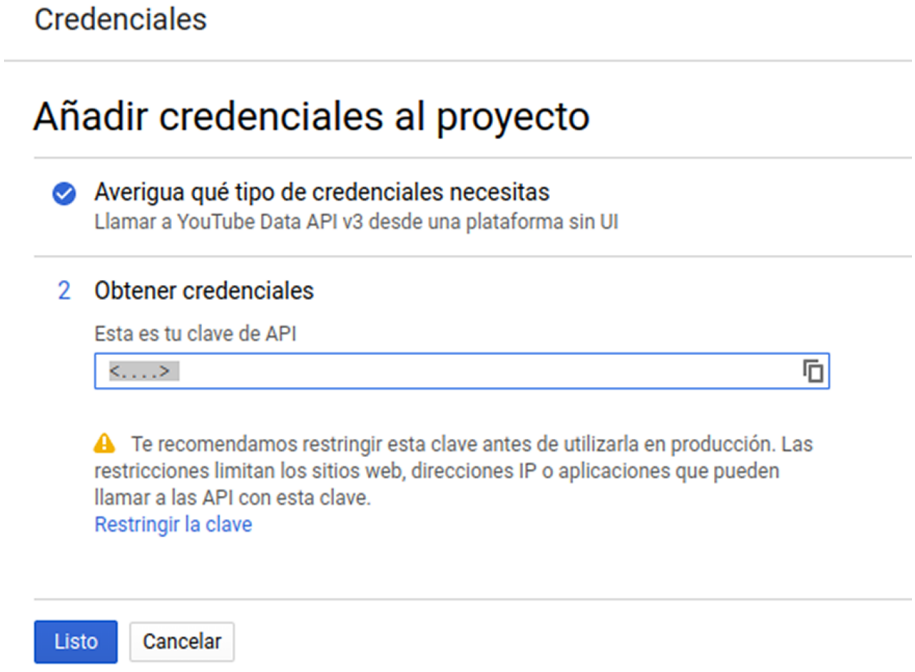
2) Creem un projecte:



3) Creem les credencials:
https://console.developers.google.com/apis/library/youtube.googleapis.com.
Fem clic al botó «Habilitar». Després a «Crear credenciales».

Seguidament, creem unes credencials per a una entitat no UI, ja que farem servir les comandes del sistema operatiu (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

Ja tenim el *token* d'accés: https://console.developers.google.com/apis/credentials

Copiarem l'*access token* i farem una petició `curl` per a comprovar que tot funciona correctament.

La petició la farem al canal dels mossos d'esquadra i la consulta serà l'obtenció de les dades 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 dels mossos:
https://www.youtube.com/channel/UC5cri6CvVLNVmF2C3GbgJTw

Resultat:


```
{
 "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-l78wL9w4xW7nIsccqpI55Pes17xfSv3KbLkE5Q=
    s88-c-k-c0xffffffff-no-rj-mo",
    "width": 88,
    "height": 88
    },
    "medium": {
    "url": "https://yt3.ggpht.com/a/AGF-l78wL9w4xW7nIsccqpI55Pes17xfSv3KbLkE5Q=
    s240-c-k-c0xffffffff-no-rj-mo",
    "width": 240,
    "height": 240
    },
    "high": {
    "url": "https://yt3.ggpht.com/a/AGF-l78wL9w4xW7nIsccqpI55Pes17xfSv3KbLkE5Q=
    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"
    }
    }
    ]
  }
```

Podem veure com l'API ens ha retornat les dades bàsiques del canal. Amb aquest resultat podem confirmar que la configuració de Google funciona correctament i ja podem utilitzar l'API per a extreure les dades.

Ara obtindrem el *data set* de la llista d'*snippets* amb la informació associada. L'API suporta un màxim de 50 resultats per consulta, per la qual cosa haurem de fer la captura de dades emprant la paginació: https://developers.google.com/youtube/v3/docs/search/list.

`pagetoken` és el paràmetre que identifica la pàgina.

La consulta tindrà la forma inicial següent:

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

I la resposta:

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

La llista d'ítems serà els 50 vídeos de la pàgina. Però volem totes les dades, per la qual cosa haurem de construir un petit *script* que faci les consultes i vagi canviant de pàgina fins que no quedin més dades a obtenir.

Un possible *script* pot ser el següent:

```
#!/bin/bash

canal=UC5cri6CvVLNVmF2C3GbgJTw
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 procés ens deixarà diversos fitxers json amb format `resultat.json.<#numero>`

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

Per a tots els fitxers, extraiem la data de publicació, el títol i la descripció de cada vídeo. Creem un csv resultant:

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

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

Normalitzarem la data en format final «dia-mes-any» i l'hora la descartarem perquè no és rellevant per a aquest cas pràctic.

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

Ara instal·larem **CSVKit**, que és un conjunt d'eines per treballar amb CSV i està disponible als repositoris de Debian i Ubuntu:

```
apt install csvkit
```

CSVKit proporciona la comanda `csvstat` que mostra les estadístiques més descriptives del fitxer csv. El processarem prèviament amb `csvcut` per a evitar errors de format, ja que en algunes línies de la descripció hi ha dobles cometes no situades correctament o amb el caràcter erroni.

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

El fitxer «stat.txt» contindrà unes estadístiques bàsiques 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
```

Amb aquesta estadística podem veure que el dia que es van publicar més vídeos va ser el 23-08-2016 amb 8 vídeos publicats.

En aquest canal es fa poc ús dels *hashtags*. Per a fer-nos una idea, els podem extreure amb l'ajuda de `sed` i `egrep` i el *hashtag* que més predomina és 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 #seguretat
     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
```

Ara, farem dues representacions visuals de les dades: la primera amb la quantitat de produccions de vídeos per any i, la segona, amb la quantitat de produccions relacionades amb les poblacions de Barcelona.

Els gràfics els farem cridant un *script* en *R* des de Bash. Per a instal·lar *R* i *ggplot2* que és el paquet que fa servir l'*script* per a construir els gràfics, els instal·larem amb:

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

En primer lloc, farem servir `csvcut` per a arreglar els problemes de format (cometes dobles mal situades) que té el fitxer, a l'igual que abans, però aquest cop guardarem els resultats a un fitxer per a poder treballar-hi de manera més còmoda.

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

Ara crearem l'esquema automàticament amb `csvsql` per a poder consultar les dades per SQL.

```
$ csvsql mossos_format.csv

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

Generem la consulta SQL per a extreure els anys i la quantitat de vídeos per any:

```
$ 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
```

Desem els resultats a un fitxer.

```
$ 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
```

L'*script* s'encarrega de fer el gràfic a partir dels arguments d'entrada:

- fitxer de dades en csv,
- coordenada X,
- coordenada Y,
- escala de l'eix X,

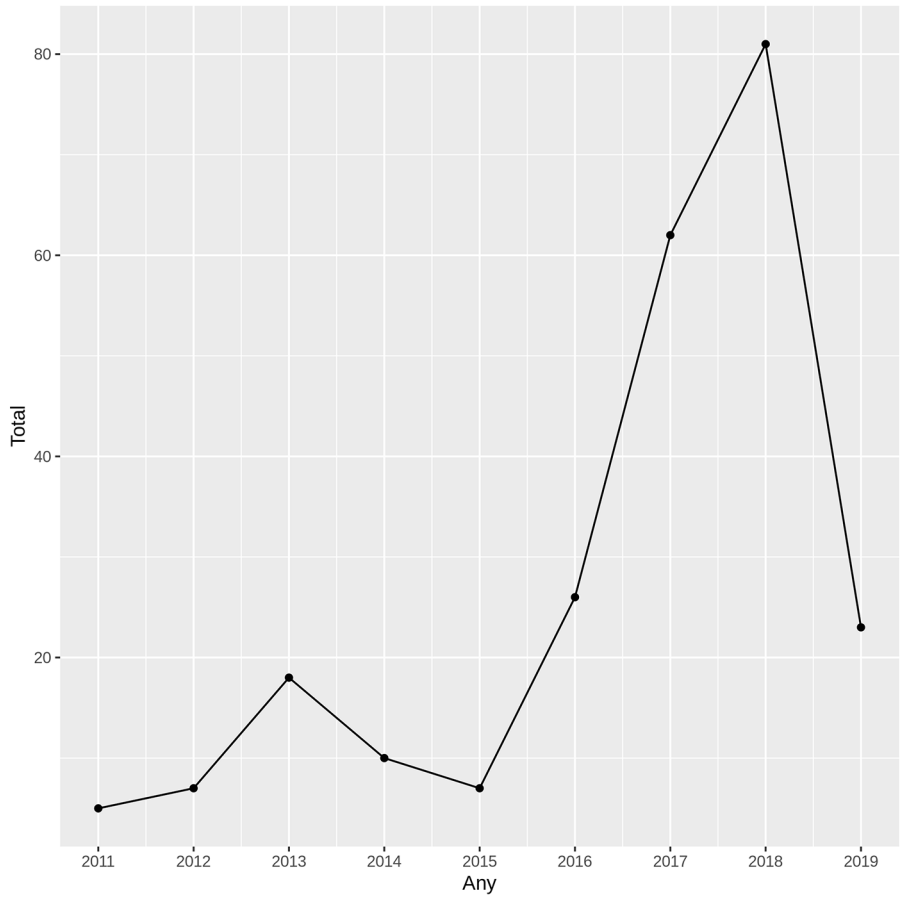
- escala de l'eix Y i
- fitxer de sortida.

```
#!/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)
```

L'executarem passant-li els arguments des de Bash:

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

L'*script* crearà un fitxer png de sortida amb la gràfica:



Vegem quants vídeos hi ha relacionats amb les poblacions de Barcelona. Per això, creuarem les dades amb les de la Wikipedia. En aquest escenari, en lloc de tirar cap a l'API de Wikipedia, obtindrem les seves dades en html i utilitzarem `xmllint`, present en la majoria de distribucions, per a fer el *parsing* de les dades que ens interessen.

L'url és:
https://es.wikipedia.org/wiki/
Anexo:Municipios_de_la_provincia_de_Barcelona

Obtindrem les poblacions de la segona columna, per la qual cosa podem fer servir l'expressió XPath següent:

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

Amb `sed` s'extreu del resultant el nom de la població, eliminant les etiquetes que no ens interessen.

La comanda final serà:

```
$ 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
```

Per a evitar problemes amb la codificació durant el filtratge, eliminarem l'accentuació:

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

Ara creuarem les dades de les poblacions i guardarem el resultat a un fitxer. Escollim afegir una columna més al csv, amb el camp `població` al principi de cada fila. També es podria resoldre creant només un fitxer a banda, amb una única columna que tingui el resultat seqüencial de cada població.

S'ha de tenir en compte que una línia pot contenir cap, una o més poblacions.

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

Quedarà:

```
"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 constituia un perill
per a les persones i l'ecosistema. L'accio, coordinada entre la Unitat ..."
<...>
```

Per a il·lustrar una altra manera diferent de compondre el fitxer resultant, en aquest exemple ho farem amb les eines de la *shell* i Bash. Però també es podria fer carregant el resultat amb `csvsql` i realitzant la consulta SQL:

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

S'extreuen les columnes i es compten les ocurrències amb `sed`:

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

O també podem utilitzar `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 fitxer resultant queda de l'estil:

```
$ 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
```

L'*script* s'encarrega de fer el gràfic a partir dels arguments d'entrada:

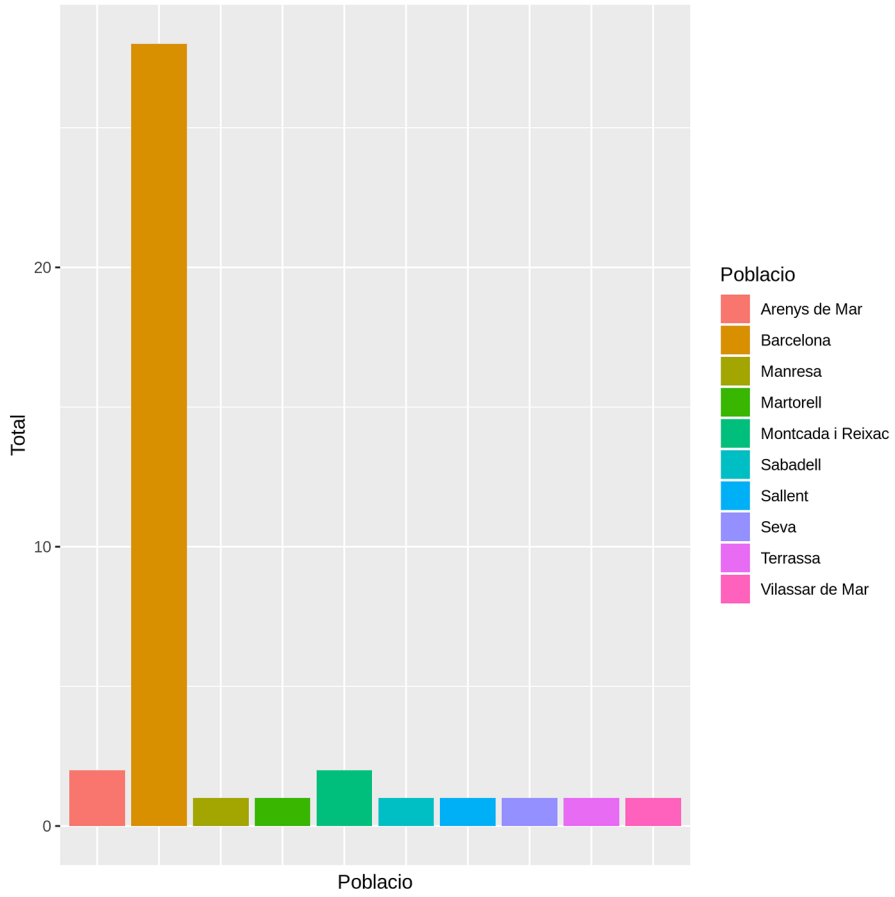
- fitxer de dades en csv,
- coordenada X,
- coordenada Y,
- emplenament i
- fitxer de sortida.

```
#!/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)
```

Executem:

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

L'*script* crearà un fitxer png de sortida amb la gràfica:


Total
20
10
0
Poblacio
Poblacio
Arenys de Mar
Barcelona
Manresa
Martorell
Montcada i Reixac
Sabadell
Sallent
Seva
Terrassa
Vilassar de Mar