PRAC1: Web Scraping

Máster Universitario en Ciencia de Datos (UOC)

Tipología y ciclo de vida de los datos

Daniel Núñez Sanz

Contexto

El año 2016 fue un año muy convulso en la política a nivel internacional. Contra todos los pronósticos y las encuestas: el Brexit gana las elecciones en Gran Bretaña, gana el no en el plebiscito sobre los acuerdos de paz en Colombia y gana Trump las elecciones. En ese momento se desconocía que había pasado o como se pudieron suceder estos tres sucesos a priori tan poco probables.

En el año 2018 aparece el escándalo de Cambridge Analytica que arrojará luz sobre estos sucesos y otros que se producirán justo después como la victoria de Bolsonaro en Brasil o el auge del populismo de derechas en Europa.

El escándalo de Cambridge Analytica vino por el uso de información de millones de usuarios de Facebook para crear anuncios políticos personalizados en las elecciones presidenciales de EE. UU. de 2016. Tras esto, uno de sus trabajadores revela que la empresa había creado una maquinaria para manipular las decisiones de los votantes. Pero ya no solo es que usaran las redes sociales para crear anuncios, sino que van más allá. En un video con cámara oculta nombran algunas tácticas que han usado; como el uso de prostitutas, sobornos o usar antiguos espías del M16 o el Mossad israelí para conseguir información de los rivales[1].

Además, crean mentiras y las difunden por redes para beneficiar a sus clientes. Pero detrás de esto hay mucho más. No solo un encargo de un partido hacia una empresa o una consultora de marketing para ganar las elecciones como hacen todos. Detrás de esta empresa hay un grupo de personas dedicadas a manipular las decisiones de los votantes en América y Europa a favor de sus intereses, en favor de una ideología concreta.

Este entramada lo podemos simplificar en una persona clave. Steve Bannon, Vicepresidente de Cambridge Analytica, asesor de la campaña de Trump en 2016 y director de Breitbart (medio de extremaderecha). Bannon ha declarado en numerosas entrevistas su deseo de crear un movimiento político populista de derechas[2]. Creó la agrupación The Movement con la idea de unir a los partidos populistas de derechas en Europa y Latinoamérica. Además de su conexión directa con Trump, Bannon ha tenido relación o ha asesorado a partidos de extremaderecha como el Frente Nacional francés, Alternativo Por Alemania, Liga Norte en Italia o a Vox en España entre muchos otros.

A pesar de que Cambridge Analytica cerró, luego crearon una nueva empresa. A parte de esta, hay más empresas que han cogido el relevo, como CLS Estrategies. Entre Cambridge Analytica,

[1] https://www.elperiodico.com/es/internacional/20180320/ejecutivos-cambridge-analyticafilmados-alardeando-prostitutas-sobornos-influencia-elecciones-channel-4-6702488

[2] https://www.eltiempo.com/mundo/eeuu-y-canada/entrevista-con-steve-bannon-sobre-el-populismo-y-el-gobierno-de-donald-trump-294662

CLS, Bannon y los partidos populistas de derechas se puede ver una estrategia común: creación de miles de bots en redes sociales, difusión de fakes news o anuncios de fraude electoral en caso de que vayan a perder las elecciones como pasó en Bolivia y el posterior Golpe de estado. Es más, lo estamos viendo en estos momentos con Trump en las actuales elecciones a la presidencia de EE. UU., pidiendo que no se tenga en cuenta el voto por correo, que se pare el recuento del voto o su recurso a la justicia y sus acusaciones de fraude ante la OEA.

El 10 de abril de 2018 Vox anuncia que se reunirá con Bannon[3] para impulsar su agenda política y establecer estrategias de propaganda de cara a las siguientes elecciones [4].

Y si nos fijamos en los resultados electoral de Vox en todos los niveles, vemos que el partido a pesar de presentarse a elecciones desde 2014, no es hasta diciembre de 2018 que logra sus primeros escaños. Tras esto, logra escaños en casi todos los parlamentos autonómicos y en el Congreso.

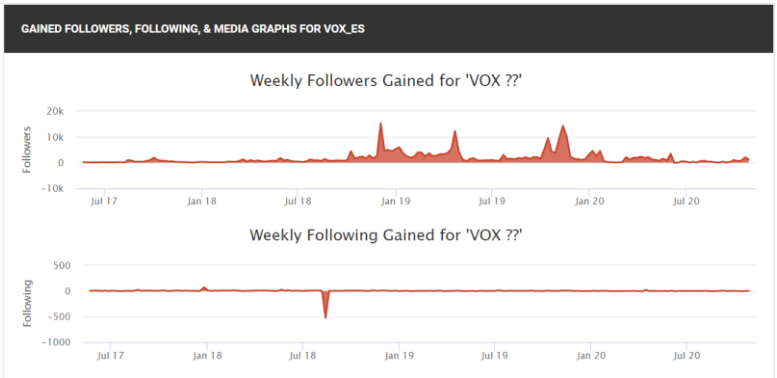
En este contexto, resulta interesante analizar las redes sociales de Vox. ¿Cómo han cambiado antes y después del encuentro de Bannon? ¿Ha tenido algún efecto el asesoramiento de Bannon en las redes sociales de Vox?

La principal red social donde se puede enviar su mensaje sin filtros y hacer más trampas sin pagar, es Twitter. Esto se debe a que Facebook, Instagram o YouTube son vistos principalmente entre quien sigue las cuentas, pero en Twitter puede lograr a poner un Hashtag en Trending Topic y lo verá todo el mundo.

Para ellos vamos a extraer el histórico de los seguidores y seguidos de la cuenta oficial de Vox en esta red social y analizarla. También cogeremos los de los otros tres principales partidos (PSOE, PP y Podemos) para facilitar un futuro análisis.

Dado que no podemos acceder al historial de los seguidores de una cuenta de Twitter, tenemos que recurrir a otra plataforma que los haya recopilado anteriormente. Esta información la podemos encontrar en Social Blade, web que proporciona información y métricas de usuarios de numerosas redes sociales, entre ellas Twitter. En el final de la web se puede ver un gráfico que tiene la información que necesitamos:

https://socialblade.com/twitter/user/vox_es/monthly



[3] Esta reunión se da gracias a Fajardí, ex del PP que se pasó a Vox, pasó por FAES y ahora está en Atlantic Councill, vinculada con CLS Estrategies.

[4] https://www.voxespana.es/noticias/bannon-el-artifice-de-la-victoria-de-trump-apuesta-por-vox-para-espana-20180410

También, indagando encontramos a otras personas que han realizado web scraping en la página de Social Blade. Por ejemplo, Anya Vastava[5] que lo hace para recopilar datos de YouTube, y en su caso recurre a Web Archive para obtener datos de hace más tiempo. En nuestro caso, lo intentamos, pero no había ninguna captura en Web Archive de los datos que necesitábamos, por lo que lo hicimos directamente desde Social Blade.

Proceso de obtención de datos

Para su obtención hemos raspado con Python la web Social Blade. Primero intentamos coger la información de la web directamente, pero nos da error. Sabe que estamos accediendo desde Python:

```
In [6]: import requests
        from bs4 import BeautifulSoup

        page = requests.get("https://socialblade.com/twitter/user/vox_es/monthly")
        soup = BeautifulSoup(page.content)
```

```
print(soup.prettify())
```

```
<html>
 <head>
  <style>
   /*!
 *  Font Awesome 4.2.0 by @davegandy - http://fontawesome.io - @fontawesome
 *  License - http://fontawesome.io/license (Font: SIL OFL 1.1, CSS: MIT License)
```


```
 */@font-face{font-family:FontAwesome;src:url(data:application/vnd.ms-fontobject;base64,xtoAAODZAAACAAIABAAAAAAAAAAAAAAAAAABA
JABAAAEAExQAAAAAAAAAAAAAAAAAAAAAEAAAAAAAAAFMQkuwAAAAAAAAAAAAAAAAAAAAAAABYARgBvAG4AdABBAHcAZQBzAG8AbQBlAAAADgBSAGUAZwB1AGwAYQ
ByAAAAJABWAGUAcgBzAGkAbwBuACAANAAuADIALgAwACAAMgAwADEAMwAAACYARgBvAG4AdABBAHcAZQBzAG8AbQBlACAAUgBlAGcAdQBsAGEAcgAAAAAAQlNHUAA
AAAAAAAAAAAAAAAAAAAADAX7AANnYANncALFgFM3pjM/SEVmjRApN5UYeAHiZoKQ+sJvend4XEADGjylbHBQxybVI0e2miS1BKUbg1dmcMfLNLtMvClM3ulUZJ+sm
YQqcO2ETIzcxEl4Thr13UpW5IKhQB7qRcpcd9W8Sz9TPYvaHUt2PNpKM1xBsFbduw19VcOEhvqjJBMr5kcViHZYKiMfUaNcsN3r0q1UT7OvUlxrw9L/ZXaAp1/5XR
hv5KA6RyVZIl88jIPmrFvFqpjLUzWweUfOQ765UJiqfsx9qjbU52OFfnvRb/yJMlg6NkNMFyBzJIGFBEHluRvvuFOyq/pP2ZQAOFNwDR2GNMUXMIGGbYu6gMKj7r9
Y4elNBisEtlJeRyMjqxrA9Nw6kRXj5mENy7KCAAtgZMDaCLBrwUn4+oBRjSTp/UyqOGmA1ChNuhShUZWZYYOMHmEDPyb5BEgkQTIKU1D2nQ6E9yuhlpzzuJ2bysLl
zSAMnGWUJaS8ieN2JINvnjlhAbA5XjCFiGZk4UoU4pipq7WHDZUZVa0yswYSlxUm7BxoTJxla1Z+jFsMBwNBA7tJJ0jNIxLq+zXAVR0jA4MzwCQKaQFlpQA8HaYCR
U6F3hTCo0I7vYqznQFhvebeae5e4Ztaq9q9o4hhVw4SzaLKKTJ0aDCoXbAuhkciGtpkgTqUxewMLowkpZQxUqYiuvjBSBRxu0BeDCeQvU9NjH1BWk+B6NiVmfYo0x
EOerIQmr4Ld8FeE+icaLJ5BuBnsiwTz9pVAUSW4+66wRshg2FRoul0cr9IzD6HbBYGIWClAyFZaPUbPWbtcJ1OiTmd4EBv0LA23kCe3zfuUmNKc3O5i8FLHqm0F6x
WywGrfD4VbtWK4FzBBHLwN3U5N9yQBm71YGW0Lu4XOWVGziJIIdrqgYeCSaVQzu+nxQ1TZqq4XI7G0zDhFRk0yKpuEsOHLKySpSdeJKT6TN+rumJY99agZK67gnWK
po3RfkTrOtD5SZkjsVbc2Ysz58cjNW62kflklLBczahXFzZZ8+ZFeZRwLQ6h2WrBgXiBIVBSlTBOpD36AWxDiXHJzIb6iSsk5SO067xHgC02TrqkaiAW1LjZA6erc
vBqbV85g+SYMoujmeZh7thicgaOPDTkWHcKzGMjK4sLbS0FRMwaj5crjiOJUycdxvaIiAkIQPIP5iZPq9CxLIhPVe6JDHIjJ0kux/c2aDNVsN2WqaEGG83qc+9h65
```



```
print(soup.prettify())
```


```
 </head>
 <body>
  <div class="cf-error-details cf-error-1020">
   <h1>
    Access denied
   </h1>
   <p>
    This website is using a security service to protect itself from online attacks.
   </p>
   <ul class="cferror_details">
    <li>
     Ray ID: 5ec5f359af5814f9
    </li>
    <li>
```

[5] https://medium.com/swlh/how-to-scrape-socialblade-for-youtube-subscription-data-ec7c4bde6933

```
print(soup.prettify())
```

```
<br/>
<ol>
 <li>
  1) You're not actually a human (i.e. a crawler/scraper)
 </li>
 <li>
  2) You're accessing the website from a computer network not typically used by humans (i.e. a data center)
 </li>
 <li>
  3) You're accessing the website from a network that has a high level of abuse (i.e. a vpn provider)
 </li>
</ol>
<br/>
If you believe you're recieving this in error please take a screen shot of this entire page and open a support ticket at
<a href="https://support.socialblade.com">
 support.socialblade.com
</a>
<br/>
If you are trying to access Social Blade from a data center via automated means you probably want to look at using our
<a href="https://socialblade.com/business-api">
```

Para solucionar este problema modificamos el user agent y algunas cabeceras HTTP:

jupyter Prac1-Webscrap Last Checkpoint: hace 5 horas (autosaved)

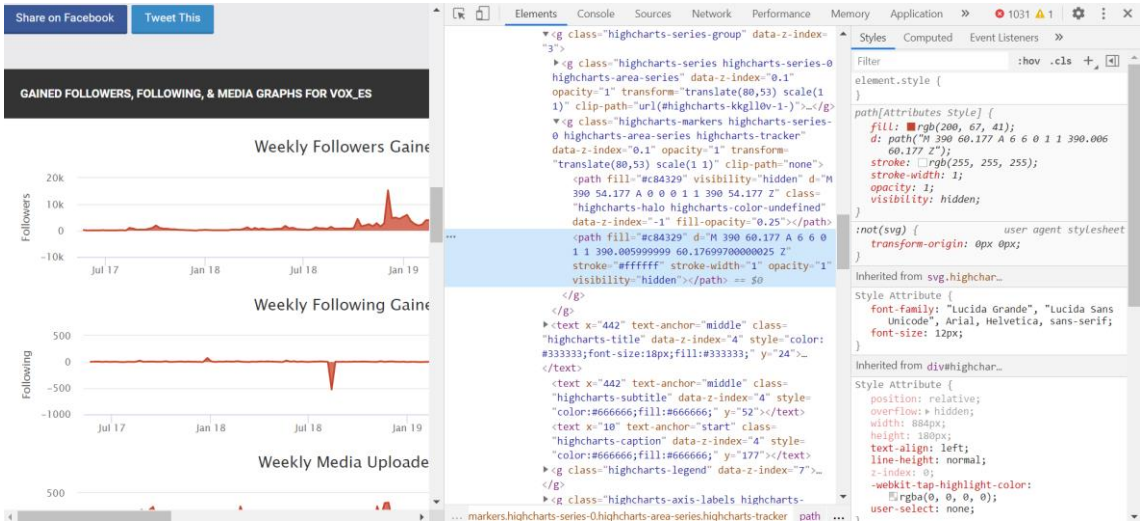
File Edit View Insert Cell Kernel Widgets Help Trusted

Run Code

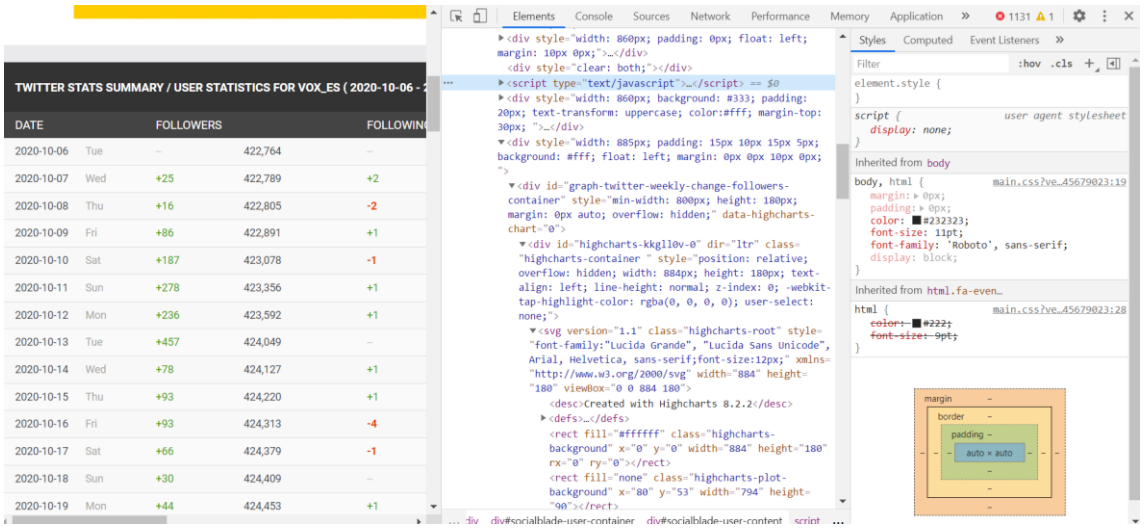
```
import time
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

%matplotlib inline
```

In [3]:
```
#Modificamos las cabeceras HTTP y el user agent al detectar la web que accedemos desde python
headers = {
"Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\
*/*;q=0.8",
"Accept-Encoding": "gzip, deflate, sdch, br",
"Accept-Language": "en-US,en;q=0.8",
"Cache-Control": "no-cache",
"dnt": "1",
"Pragma": "no-cache",
"Upgrade-Insecure-Requests": "1",
"User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_3) AppleWebKit/5\
37.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36"
}
```

Analizamos la web:



Vemos donde está la información que queremos y que su código es en JavaScript



Tras esto, cogemos con Beautifulsoup la información de la web indicando es de tipo JavaScript

```
# Tomamos la informacion de la web
soup = requests.get("https://socialblade.com/twitter/user/vox_es/monthly", headers=headers)
```

```
soup = bs(soup.text, 'lxml')
script_divs = soup.find_all('script', {'type': 'text/javascript'})
```

soup

```
<!DOCTYPE html>\n<html><head>\n<title>VOX ??'s Twitter Monthly Stats (Social Blade Twitter Statistics)</title>\n<meta content
="VOX ?? Twitter Statistics and Summary Page. Discover daily Twitter statistics, VOX ?? ranking charts, and more! Provided by
SocialBlade.com" name="description"/>\n<meta content="VOX ??, Twitter Summary Profile, Twitter Followers, Twitter Statistics,
Social Blade, Socialblade, Socialblade Stats, Twitter Stats, Twitter" name="keywords"/>\n<meta charset="unicode-escape"/>\n<m
eta content="website" property="og:type"/>\n<meta content="VOX ?? Twitter Statistics and Summary Page. Discover daily Twitter
statistics, VOX ?? ranking charts, and more! Provided by SocialBlade.com" property="og:description"/>\n<meta content="summary
_large_image" name="twitter:card"/>\n<meta content="socialblade" name="twitter:site"/>\n<meta content="VOX ?? Twitter Statist
ics and Summary Page. Discover daily Twitter statistics, VOX ?? ranking charts, and more! Provided by SocialBlade.com" name
="twitter:description"/>\n<meta content="//socialblade.com/images/share-badge.png" property="og:image"/>\n<link href="http
s://socialblade.com/images/share-badge.png" rel="image_src"/>\n<meta content="https://socialblade.com/images/SocialbladeBadg
e.png" name="twitter:image"/>\n<meta content="VOX ??'s Twitter Monthly Stats (Social Blade Twitter Statistics)" name="twitte
r:title"/>\n<meta content="urgo" name="twitter:creator"/>\n<meta content="dmisdtmthe6pj4frg" name="dailymotion-domain-verific
ation"/>\n<script type="text/javascript">var _sf_startpt=(new Date()).getTime()</script>\n<script type="text/javascript">var
beforeload = (new Date()).getTime()</script>\n<link href="/favicon.ico" rel="shortcut icon" type="image/x-icon"/>\n<link href
="/apple-touch-icon.png" rel="apple-touch-icon"/>\n<link href="/apple-touch-icon-57x57.png" rel="apple-touch-icon" sizes="57x
57"/>\n<link href="/apple-touch-icon-72x72.png" rel="apple-touch-icon" sizes="72x72"/>\n<link href="/apple-touch-icon-76x76.p
ng" rel="apple-touch-icon" sizes="76x76"/>\n<link href="/apple-touch-icon-114x114.png" rel="apple-touch-icon" sizes="114x11
4"/>\n<link href="/apple-touch-icon-120x120.png" rel="apple-touch-icon" sizes="120x120"/>\n<link href="/apple-touch-icon-144x
144.png" rel="apple-touch-icon" sizes="144x144"/>\n<link href="/apple-touch-icon-152x152.png" rel="apple-touch-icon" sizes="1
```

script_divs

```
[<script type="text/javascript">var _sf_startpt=(new Date()).getTime()</script>,
 <script type="text/javascript">var beforeload = (new Date()).getTime()</script>,
 <script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js" type="text/javascript"></script>,
 <script src="/js/cookie.js?version=9001.8908908097" type="text/javascript"></script>,
 <script src="//cdnjs.cloudflare.com/ajax/libs/dygraph/1.0.1/dygraph-combined.js" type="text/javascript"></script>,
 <script type="text/javascript">\n\t\n\t    $(function () { \n\n      \n            Highcharts.chart('graph-twitter-weekly-ch
ange-followers-container', {\n            chart: { type: 'area', zoomType: 'xy', resetZoomButton: { position: { x: 0,  y: -52
} } },  \n            credits: { enabled: false },\t\t    \n            title: { text: 'Weekly Followers Gained for \\'VOX ??
\\'' },\n            xAxis: {\n                //type: 'datetime', labels: { overflow: 'justify' }\n                    typ
e: 'datetime',\n                    dateTimeLabelFormats: { // don't display the dummy year\n                        month:
'%b %y',\n                    year: '%b'\n                    },\n                    title: {\n                        t
ext: ''\n                    },\n                    events: {\n                        afterSetExtremes: function (e) {\n
if(e.trigger == "zoom") {\n                            this.chart.update({\n                                toolti
p:\n                                {\n                                    enabled: true\n
}\n                                });\n                                }\n                                }\n                    }
\n            },\n            yAxis: { title: { text: 'Followers' }, minorGridLineWidth: 1, gridLineWidth: 1, alternateG
ridColor: null, },\n            tooltip: { valueSuffix: ' Followers', hideDelay: '50',\n                formatter: function()
{\n                var x_date = this.x;\n                var last_x_date = this.x + 604800000;\n                v
ar dateform = '<span style = "font-size: 5pt;">Week of ' + Highcharts.dateFormat('%b %e, %Y', new Date(this.x)) + ' - ' +  Hi
```

Ahora definimos una función en la que tomaremos los datos de la web y la depuraremos para tener una lista. Se explica paso a paso el proceso de la función:

Primero cogemos solo la fila 5 que es donde está la información que queremos.

```
script_divs[5]
```

```
abelHeight, point) {\n                    var tooltipX = point.plotX + 20;\n                    var tooltipY = 0;\n\n
if((labelWidth + point.plotX) > 800) {\n                        tooltipX = (800 - labelWidth + 80);\n                    }\n
\n                    return {\n                        x: tooltipX,\n                        y: tooltipY\n
};\n                } \n            },\n            plotOptions: { \n            spline: { \n                lineWidth: 2,\n
states: {\n                hover: { lineWidth: 3 }\n                }\n            }\n            },\n            series: [{
name: '', showInLegend: false, animation: false, color: '#c84329', lineWidth: 2, data: [[1495425600000,209],[1496030400000,6
9],[1496635200000,107],[1497240000000,105],[1497844800000,97],[1498449600000,165],[1499054400000,111],[1499659200000,109],[15
00264000000,106],[1500868800000,138],[1501473600000,177],[1502164800000,89],[1502683200000,1027],[1503288000000,791],[1503892
800000,337],[1504497600000,406],[1505188800000,357],[1505707200000,665],[1506312000000,1017],[1506916800000,1955],[1507521600
000,1022],[1508126400000,739],[1508731200000,735],[1509336000000,481],[1509944400000,508],[1510549200000,295],[1511154000000,
266],[1511758800000,200],[1512363600000,162],[1513400400000,22],[1513573200000,182],[1514178000000,220],[1514782800000,306],
[1515387600000,209],[1515992400000,186],[1516597200000,141],[1517202000000,177],[1517806800000,168],[1518411600000,158],[1519
016400000,406],[1519621200000,390],[1520226000000,321],[1520827200000,672],[1521432000000,1258],[1522123200000,384],[15226416
00000,1036],[1523246400000,511],[1523851200000,876],[1524456000000,520],[1525060800000,428],[1525665600000,650],[152627040000
0,753],[1526875200000,744],[1527480000000,1801],[1528084800000,879],[1528689600000,1080],[1529294400000,583],[1529899200000,4
73],[1530504000000,401],[1531108800000,274],[1531713600000,481],[1532318400000,1232],[1532923200000,896],[1533528000000,910],
[1534132800000,706],[1534737600000,1436],[1535342400000,724],[1535947200000,718],[1536552000000,835],[1537156800000,835],[153
7761600000,760],[1538366400000,1003],[1538971200000,4467],[1539576000000,1647],[1540180800000,2012],[1540785600000,2451],[154
1394000000,1646],[1541998800000,2854],[1542603600000,1567],[1543208400000,2781],[1543813200000,15320],[1544418000000,4757],[1
545022800000,4972],[1545627600000,4438],[1546232400000,5245],[1546837200000,6001],[1547442000000,3653],[1548046800000,2528],
```

La separamos por “[“ ya que es el inicio de cada uno de los datos que vamos a coger.

```
cadena_sep = str(cadena_ori).split('[')
cadena_sep
```

```
er: function(labelWidth, labelHeight, point) {\n                    var tooltipX = point.plotX + 20;\n                    var
tooltipY = 0;\n\n                    if((labelWidth + point.plotX) > 800) {\n                        tooltipX = (800 - labelW
idth + 80);\n                    }\n\n                    return {\n                        x: tooltipX,\n
y: tooltipY\n                    };\n                } \n            },\n            plotOptions: { \n            spline: {
\n                lineWidth: 2,\n                states: {\n                hover: { lineWidth: 3 }\n                }\n
}\n            },\n            series: ',
 "{ name: '', showInLegend: false, animation: false, color: '#c84329', lineWidth: 2, data: ",
 '',
 '1495425600000,209],',
 '1496030400000,69],',
 '1496635200000,107],',
 '1497240000000,105],',
 '1497844800000,97],',
 '1498449600000,165],',
 '1499054400000,111],',
 '1499659200000,109],',
 '1500264000000,106],',
 '1500868800000,138],',
 '1501473600000,177],',
 '1502164800000,89],',
```

Separamos las cadenas por las filas que se indicaran en la función que son el principio y el final de los datos

```
lista = cadena_sep[3: 182]
lista
```

```
['1495425600000,209],',
 '1496030400000,69],',
 '1496635200000,107],',
 '1497240000000,105],',
 '1497844800000,97],',
 '1498449600000,165],',
 '1499054400000,111],',
 '1499659200000,109],',
 '1500264000000,106],',
 '1500868800000,138],',
 '1501473600000,177],',
 '1502164800000,89],',
 '1502683200000,1027],',
 '1503288000000,791],',
 '1503892800000,337],',
 '1504497600000,406],',
 '1505188800000,357],',
 '1505707200000,665],',
 '1506312000000,1017],',
```

y por último quitamos “],” del final de cada fila.

```
lista = [test.replace('],', '').replace('"', '') for test in lista]
lista
```

```
['1495425600000,209',
 '1496030400000,69',
 '1496635200000,107',
 '1497240000000,105',
 '1497844800000,97',
 '1498449600000,165',
 '1499054400000,111',
 '1499659200000,109',
 '1500264000000,106',
 '1500868800000,138',
 '1501473600000,177',
 '1502164800000,89',
 '1502683200000,1027',
 '1503288000000,791',
 '1503892800000,337',
 '1504497600000,406',
 '1505188800000,357',
 '1505707200000,665',
 '1506312000000,1017',
```

Con esta función obtendremos una lista con la fecha y separado por una coma, el dato según las filas que se pongan.

Función final:

```
# Definimos una funcion para coger solo la parte que nos interesa
def lista_scrap(x, y):
    # Nos quedamos solo con la fila 5 de la lista ya que es donde esta la ifnormacion que queremos
    cadena_ori = script_divs[5]
    # Creamos una lista separando por [ que es por donde empieza nuestra informacion
    cadena_sep = str(cadena_ori).split('[')
    # Selecionamos las filas que queremos segun si es seguidores, menos seguidores...
    lista = cadena_sep[x:y]
    # eliminamos ,]
    lista = [test.replace('],', '').replace('"', '') for test in lista]
    return lista
```

Ahora hacemos otra función para depurar la lista y hacer un data frame. En ella usamos la función anterior para obtener la lista. Explicamos paso a paso la función:

Creamos un data frame separando la lista por comas. Indicando que la posición 0 teniendo en cuenta la coma como separación, obtenemos la Semana…

```
df = pd.DataFrame()
df['Semana'] = [x.split(',')[0] for x in lista]
df
```

	Semana
0	1495425600000
1	1496030400000
2	1496635200000
3	1497240000000
4	1497844800000
5	1498449600000
6	1499054400000
7	1499659200000
8	1500264000000

y la posición 1 son los seguidores.

```
df['Seguidores'] = [x.split(',')[1] for x in lista]
df
```

	Semana	Seguidores
0	1495425600000	209
1	1496030400000	69
2	1496635200000	107
3	1497240000000	105
4	1497844800000	97
5	1498449600000	165
6	1499054400000	111

Además, tenemos que depurar el formato de la fecha que está en epoch con tres ceros. Primero le quitamos los tres ceros, lo pasamos a numérico y hacemos un bucle para pasarlo a un formato de fecha normal.

```
df['Semana'] = df['Semana'].map(lambda x: str(x)[:-3])
df['Semana'] = pd.to_numeric(df['Semana'])
for n in range(len(df)):
    df['Semana'][n] = datetime.datetime.fromtimestamp(df['Semana'][n]).strftime('%d-%m-%Y')
df
```

```
C:\Users\Dany\Anaconda2\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.htm
  after removing the cwd from sys.path.
C:\Users\Dany\Anaconda2\lib\site-packages\pandas\core\indexing.py:190: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.htm
  self._setitem_with_indexer(indexer, value)
```

	Semana	Seguidores
0	22-05-2017	209
1	29-05-2017	69
2	05-06-2017	107
3	12-06-2017	105
4	19-06-2017	97

Por último, obtenemos el data frame depurado. Función final es la siguiente:

```
# Hacemos otra funacion para crear un dataframe con los datos de la anterior funcion
def lista_dataf(x, y):
    df = pd.DataFrame()
    #Usamos la funcion anterior para coger los datos
    lista = lista_scrap(x, y)
    # Señalamos la primera columna como la fecha y sacamos los datos de la lista
    df['Semana'] = [x.split(',')[0] for x in lista]
    # Señalamos la segunda como los seguidores
    df['Seguidores'] = [x.split(',')[1] for x in lista]
    #Ahora hay que pasar la fecha a un formao mas legible
    #Convertir epoch a fecha normal
    # tenemos que eliminar los tres ultimos ceros para pasorlo
    df['Semana'] = df['Semana'].map(lambda x: str(x)[:-3])
    #pasamos la columna a numeros
    df['Semana'] = pd.to_numeric(df['Semana'])
    #pasamos a formato fecha
    for n in range(len(df)):
        df['Semana'][n] = datetime.datetime.fromtimestamp(df['Semana'][n]).strftime('%d-%m-%Y')
    return df
```

Ahora inspeccionamos la descarga de datos que hemos hecho en el scraping y vemos la fila 5 separándola por "[" obteniendo una lista. Si analizamos los datos obtenidos vemos que de la fila 3 a la 182 son los datos de los nuevos seguidores, de la 185 a la 364 son los seguidos y de la 367 a la 546 es la media. Con esto ya podemos aplicar la función a cada una de las filas para obtener un data frame para cada uno de los datos.

```
# Si inspeccionamos los datos vemos de donde a donde van cada uno de los graficos
cadena_ori = script_divs[5]
cadena_sep = str(cadena_ori).split('[')

seguidores_ganados = lista_dataf(3, 182)
seguidos_nuevos = lista_dataf(185, 364)
media = lista_dataf(367, 546)
```

Ahora creamos el data frame definitivo para un partido juntando los datos de las tres bases.

```
# Juantamos las tres variables
Base_conjunta = pd.DataFrame()
Base_conjunta['Semana'] = seguidores_ganados['Semana']
Base_conjunta['Seguidores'] = seguidores_ganados['Seguidores']
Base_conjunta['Seguidos'] = seguidos_nuevos['Seguidores']
Base_conjunta['Media'] = media['Seguidores']

#Vemos la cabecera de la base
Base_conjunta.head(5)
```

	Semana	Seguidores	Seguidos	Media
0	22-05-2017	209	5	113
1	29-05-2017	69	7	84
2	05-06-2017	107	9	121
3	12-06-2017	105	2	71
4	19-06-2017	97	8	127

Por último, guardamos la base:

```
# Guardamos la base en csv
Vox_Twitter = Base_conjunta
Vox_Twitter.to_csv('Vox_Twitter.csv', index=False)
```

Ahora tenemos que hacer el mismo proceso con el PP, PSOE y Podemos para tener la base completa con los cuatro partidos más relevantes y así poder comparar.

Se decide eliminar la media ya que no parece aportar mucha información.

```
# PSOE
soup = requests.get("https://socialblade.com/twitter/user/PSOE/monthly", headers=headers)
soup = bs(soup.text, 'lxml')
script_divs = soup.find_all('script', {'type': 'text/javascript'})

seguidores_ganados = lista_dataf(3, 230)
seguidos_nuevos = lista_dataf(233, 460)
media = lista_dataf(463, 690)

Base_conjunta = pd.DataFrame()
Base_conjunta['Semana'] = seguidores_ganados['Semana']
Base_conjunta['Seguidores'] = seguidores_ganados['Seguidores']
Base_conjunta['Seguidos'] = seguidos_nuevos['Seguidores']
Base_conjunta['Media'] = media['Seguidores']

PSOE_Twitter = Base_conjunta

PSOE_Twitter.to_csv('PSOE_Twitter.csv', index=False)
```

Una vez aplicado a los cuatro partidos, solo tenemos que juntar las cuatro bases y guardarla

```
# Juntamos las cuatro bases y dejamos solo seguidores y seguidos
Partidos_Twitter = pd.DataFrame()

# Eliminamos la media ya que no nos aporta
PSOE_Twitter.drop(['Media'], axis = 'columns', inplace=True)
Podemos_Twitter.drop(['Media'], axis = 'columns', inplace=True)
Vox_Twitter.drop(['Media'], axis = 'columns', inplace=True)
PP_Twitter.drop(['Media'], axis = 'columns', inplace=True)

# Al juntarlos vemos que hay algunas semanas que no coincide exactamente y no se juntan bien.
PSOE_Twitter.Semana[77] = '11-12-2017'
PP_Twitter.Semana[58] = '11-12-2017'
Vox_Twitter.Semana[11] = '07-08-2017'
Vox_Twitter.Semana[16] = '11-09-2017'
Vox_Twitter.Semana[29] = '18-12-2017'
Vox_Twitter.Semana[44] = '26-03-2018'

# Juntamos las bases en una conjunta
Partidos_Twitter = pd.merge(PSOE_Twitter, Podemos_Twitter, on='Semana', how='outer', suffixes=( '_PSOE', '_Podemos'))
Partidos_Twitter = pd.merge(Partidos_Twitter, PP_Twitter, on='Semana', how='outer', suffixes=( '_prueba', '_PP'))
Partidos_Twitter = pd.merge(Partidos_Twitter, Vox_Twitter, on='Semana', how='outer', suffixes=( '_PP', '_Vox'))

Partidos_Twitter.to_csv('Partidos_Politicos_Twitter.csv', index=False)
```

Dataset

Al dataset se le da el nombre de “Partidos_Politicos_Twitter” y queda de la forma siguiente. Se muestra el final ya que en el inicio solo hay datos del PSOE que es el único partido que tiene datos desde 2016, mientras que el resto empiezan a tener datos en 2017.

```
Partidos_Twitter.tail(5)
```

	Semana	Seguidores_PSOE	Seguidos_PSOE	Seguidores_Podemos	Seguidos_Podemos	Seguidores_PP	Seguidos_PP	Seguidores_Vox	Seguidos_Vox
223	21-09-2020	315	-3	224	-1	3277	2671	255	2
224	28-09-2020	355	-5	84	2	409	-11	1021	4
225	05-10-2020	435	-6	454	2	613	-2	623	0
226	12-10-2020	472	-1	493	3	652	-33	817	-3
227	19-10-2020	988	-6	992	6	-387	-234	2048	-1

El data set está compuesto por nueve variables:

- Semana que hace referencia a la fecha semanal de los datos de las otras variables.
- Seguidores del partido (hay una por cada partido), que son el número de nuevos seguidores que ha conseguido el partido esa semana.
- Seguidos del partido (hay una por cada partido), que indica el número de cuentas nuevas que sigue el partido en Twitter, este puede ser negativo en caso de dejar de seguir cuentas.

El data set contiene los datos del PSOE, PP, Podemos y Vox. En el momento del Scraping, la base tiene 228 filas. Temporalmente, comprenden desde junio de 2016 hasta octubre de 2020 para el PSOE, y desde mayo de 2017 para el resto de los partidos.

Como último detalle, publicamos el data set en Zanodo con una licencia de **Creative Commons Attribution 4.0 International** ya que los datos en si son de dominio público y no somos quien para poner otra licencia que no sea libre. También es importante que los datos sean públicos para una mayor difusión de conocimiento.

https://zenodo.org/record/4247917#.X6RguVCCHcd

Publication date:
November 5, 2020
DOI:
DOI 10.5281/zenodo.4247917
Keyword(s):
politica, partidos politicos, Twitter
License (for files):
Creative Commons Attribution 4.0 International

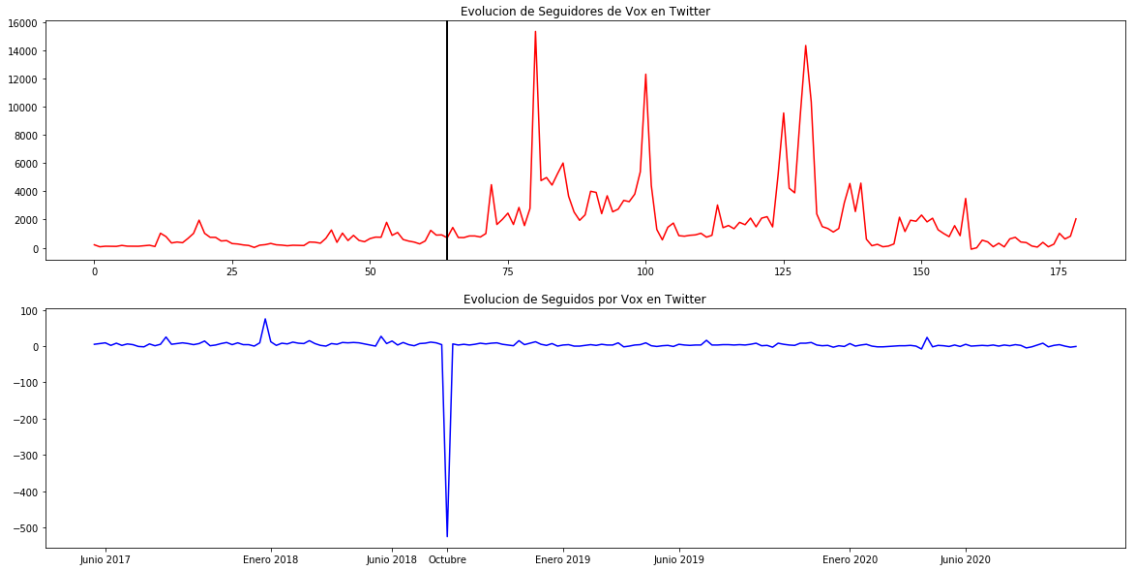
También cuando subimos el código y la base a Github le hemos puesto una licencia de **GNU General Public License v3.0**

https://github.com/dnunezs/Partidos_Politicos_Twitter

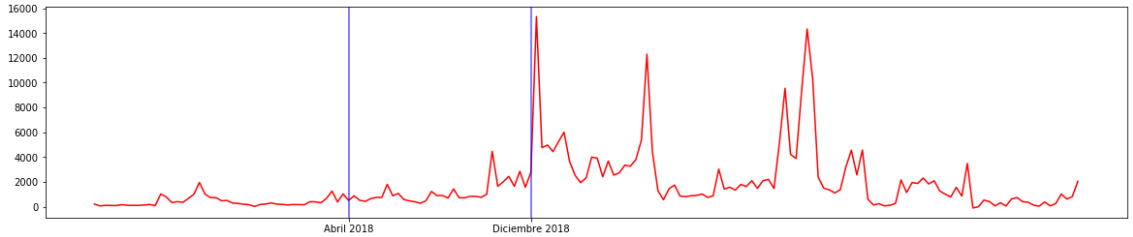
Interés del conjunto de datos y análisis preliminar.

De un primer análisis podemos sacar las siguientes conclusiones que responden a las preguntas planteadas al principio.

Para ello, lo vamos a hacer con gráficos (el código de los gráficos está en otro archivo en GitHub). En el primero se puede ver la evolución histórica de los seguidores y seguidos de la cuenta oficial de Vox en Twitter desde 2017 hasta ahora. Está marcado con una línea negra un pico descendiente que tienen de cuentas que sigue Vox en octubre de 2018. Recordemos que la primera reunión de Vox y Bannon se produjo en abril de 2018. Antes de esto no hay movimientos reseñables ni en los seguidores ni en los seguidos. Y el único movimiento que vemos grande es que la cuenta deja de seguir a 500 cuentas en una misma semana de octubre de 2018.



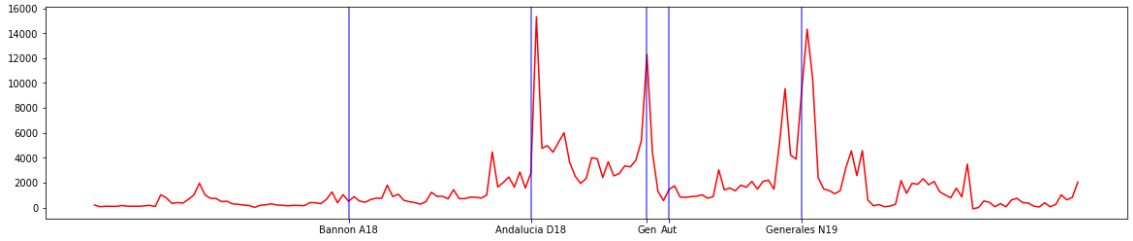
Ahora nos centraremos en los nuevos seguidores. Marcamos en el gráfico una línea en la semana del 10 abril de 2018 que es cuando se hizo pública la primera reunión con Bannon y otra en la semana del 2 de diciembre de 2018 que son las elecciones al Parlamento andaluz, momento en el cual Vox logra sus primeros escaños.



Desde 2017 hasta la llegada oficial de Bannon (abril 2018) no hay muchas variaciones en los seguidores que tiene Vox. Es desde marzo de 2018, cuando Vox empieza a tener picos de seguidores más continuos.

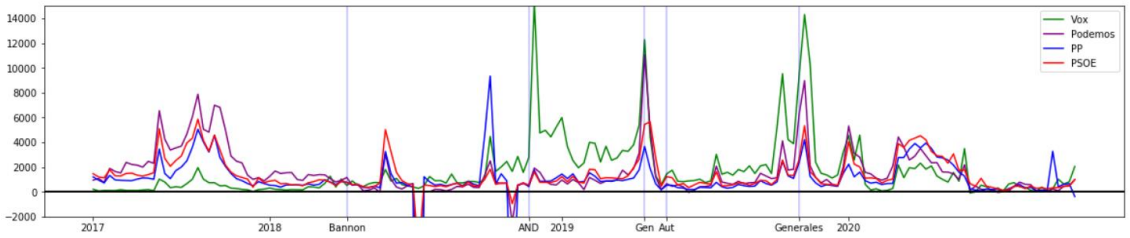
Se aprecia que, desde la llegada de Bannon hasta las elecciones en Andalucía, la cuenta de Vox empieza a tener muchos más nuevos seguidores semanales. Esto implica que, o por un lado Vox consigue en ese periodo llegar a la gente, ser conocidos, que se interesen por ellos y les empiecen a seguir, o por otro, que parte de la estrategia de Bannon sea crear miles de cuentas para difundir cuentas como las de Vox y su mensaje. Así, ampliar un mensaje y aparentar que tiene muchos seguidores. Lo más probable es que haya sido una combinación de las dos técnicas.

Nos podemos fijar más, poniendo el punto de mira en las elecciones, ya que es lógico que haya picos altos tras unas elecciones donde ganen escaños y salgan en las televisiones y les sigan.



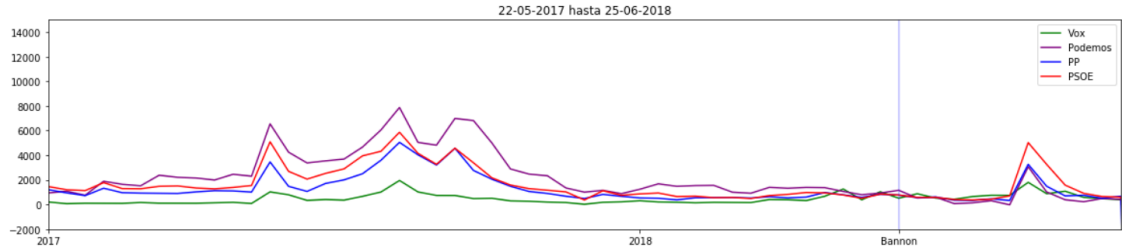
Si nos fijamos, los picos más altos se producen la misma o la siguiente semana de unas elecciones, algo que puede ser normal por la difusión en los medios.

Para analizar estos picos vamos a compararlo con el resto de los partidos. Si todos los partidos tienen picos en común, quiere decir que tienen una causa común: campaña electoral o elecciones.

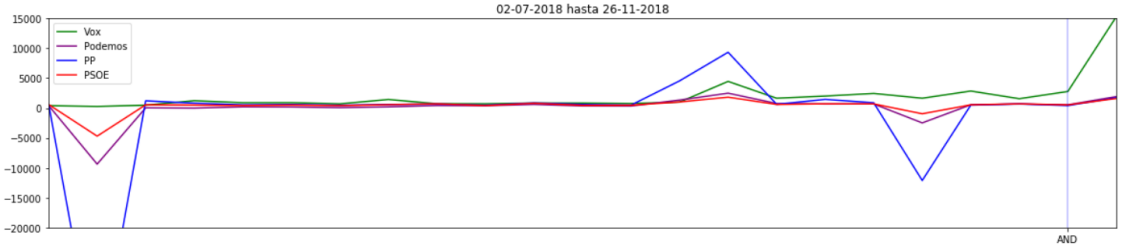


Con rasgos generales se puede ver una evolución muy similar en los cuatro partidos. Las subidas y bajadas son muy parecidas. Pero hay dos cosas que llaman la atención. La primera, desde el principio de los datos hasta un poco antes de las elecciones en Andalucía, la evolución de Vox es similar o inferior a la del resto de partidos, pero después y hasta 2020 tiene una evolución muy superior al resto. La segunda, es que hay periodos concretos donde Vox rompe la tendencia del resto de partidos.

Hagamos zoom para verlo mejor. Primero vemos desde 22-05-2017 hasta 25-06-2018:

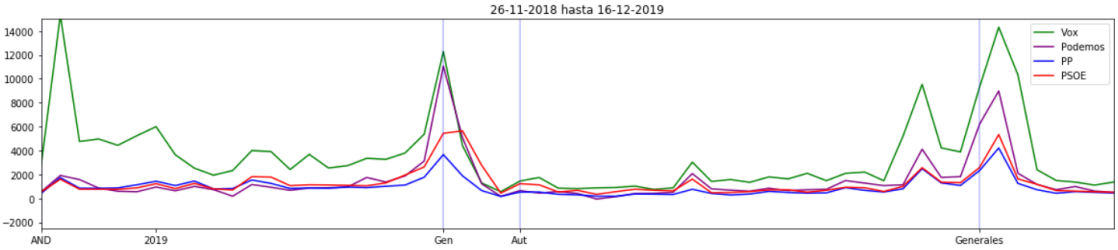


En este periodo la evolución es la misma en todos los partidos. Las subidas y bajadas siguen el mismo patrón. Pero desde entonces empiezan a haber diferencia con Vox. Vemos desde 02-07-2018 hasta 26-11-2018 02-07-2018 hasta 26-11-2018 justo el periodo anterior a las elecciones en Andalucía.



Aquí se ve como hay dos puntos donde todos los partidos pierden seguidores salvo Vox que no gana muchos, pero nunca llega a perder seguidores.

Por último, veamos el periodo desde 26-11-2018 hasta 16-12-2019 que comprende desde las elecciones andaluzas hasta las Generales de noviembre de 2019. Un periodo con muchas elecciones.



Podemos ver como la evolución de Vox es la misma que el resto de los partidos. Tienen las mismas subidas y bajadas de nuevos seguidores. Solo hay una diferencia significativa, la evolución de Vox es muy superior a la des resto de partidos justo antes de las elecciones y las semanas posteriores. En el periodo entre las andaluzas y justo antes de las primeras generales de 2019 hablamos de una media de 3.000 seguidores más a la semana que el partido que más gane cada semana, un total de 61.000 seguidores más que el resto.

Lo mismo ocurre los dos meses antes de las elecciones generales de noviembre de 2019. Vox gana una media de 2.500 nuevos seguidores cada semana más que el partido que más seguidores consigue. En total suma 26.000 nuevos seguidores más que el resto de los partidos.

En conclusión, aunque en un análisis preliminar si parece haber ciertos datos inflados artificialmente en comparación con el resto de los partidos, es imposible afirmarlo categóricamente. La única forma de comprobarlo seria analizar todos sus seguidores y ver hasta que punto hay bots o cuentas artificiales que no correspondan con una persona real de a pie.

Autoría

El trabajo se ha realizado individualmente por Daniel Núñez Sanz.

Contribuciones	Firma
Investigación previa	DNS
Redacción de las respuestas	DNS
Desarrollo código	DNS