Facultat de Ciències de la Comunicació

Treball de fi de grau

Títol

Periodisme de dades:
Anàlisi del discurs dels partits polítics a Twitter durant la campanya electoral de les eleccions generals espanyoles de 2015

Autor/a

Tristan Ibáñez Sabran

Tutor/a

Marta Sabater Casals

Departament	Departament de Mitjans, Comunicació i Cultura
Grau	Periodisme
Tipus de TFG	Projecte
Data	03/06/2016

Universitat Autònoma de Barcelona

Full resum del TFG

Títol del Treball Fi de Grau:

Català:

Periodisme de dades:
Anàlisi del discurs dels partits polítics a Twitter durant la campanya electoral de les eleccions generals espanyoles de 2015

Castellà: Periodismo de datos: análisis del discurso de partidos políticos en Twitter durante la campaña electoral de las elecciones generales españolas de 2015

Anglès: Data journalism: discourse analysis of political parties through Twitter during 2015 Spanish general election

Autor/a: Tristan Ibáñez Sabran

Tutor/a: Marta Sabater Casals

Curs: 2015/16

Grau: Periodisme

Paraules clau (mínim 3)

Català: periodisme, dades, programació, visualització, anàlisi, discurs

Castellà: periodismo, datos, programación, visualización, análisis, discurso

Anglès: journalism, data, programming, visualization, analysis, discourse

Resum del Treball Fi de Grau (extensió màxima 100 paraules)

Català: El treball consisteix en l'elaboració d'una peça informativa que inclou contingut extret i expressat a través de les rutines productives pròpies del periodisme de dades. Això inclou obtenció, anàlisi i visualització d'un conjunt de dades, en aquest cas una base de 13.520 piulades a Twitter dels quatre grans partits polítics i els seus líders, extrets durant la campanya electoral de les eleccions generals espanyoles de 2015, i l'explicació de cada una d'aquestes fases. Inclou una explicació conceptual, contextual i històrica del periodisme de dades i un anàlisi dels recursos productius necessaris per dur a terme la peça.

Castellà: El trabajo consiste en la elaboración de una pieza informativa que incluye contenido extraído y expresado a través de las rutinas productivas propias del periodismo de datos. Esto incluye la obtención, análisis y visualización de un conjunto de datos, en este caso una base de 13.520 mensajes de Twitter de los cuatro partidos políticos y sus líderes, extraidos durante la campaña electoral de las elecciones generales españolas de 2015, y la explicación de cada una de estas fases. Incluye una explicación conceptual, contextual e histórica del periodismo de datos y un análisis de los recursos productivos necesarios para llevar a cabo la pieza.

Anglès: The project involves the development of a journalistic piece that includes content extracted and expressed through the productive routines of data journalism. This includes the gathering, analysis and visualization of a data set of 13,520 tweets of the four political parties and their leaders, extracted during the election campaign of the Spanish general elections of 2015, and the explanation of each of these phases. It includes an explanation about data journalism from a conceptual, contextual and historical point of view, and an analysis of the productive resources needed to produce the piece.

Índex

1 Introducció

Aquest treball consisteix en un exercici pràctic de periodisme de dades. Té com a objectiu dur a terme una informació obtinguda i presentada a través de les rutines productives del periodisme de dades, i analitzar i explicar les diferents parts en què es divideixen aquestes rutines. Addicionalment, aprofitant la dissecció que hem fet del procés de confecció de la peça, proposem un petit estudi de viabilitat empresarial que analitza i presenta de manera detallada la tasca que portaria a terme un periodista de dades en l'entorn d'un mitjà de comunicació convencional de premsa escrita a Internet.

Per començar, presentem una explicació del que és el periodisme de dades. Explicarem quin és el seu origen, i quin és el motiu de la seva expansió actual als principals mitjans de comunicació internacionals. També presentarem el procés de producció que s'associa a aquesta disciplina, i quines són les habilitats necessàries dels membres que conformen un equip tipus de periodisme de dades. Finalment, analitzarem quina és la seva situació en els mitjans de comunicació del nostre entorn.

Pel que fa a l'exercici, procedirem a confeccionar una anàlisi del llenguatge emprat pels partits polítics i els seus líders a la plataforma de xarxa social Twitter, durant el període de campanya de les eleccions generals del passat 20 de desembre de 2016. Explicarem totes les passes en què es divideix el procés, quant temps s'ha destinat a cada part i amb quins recursos, i justificarem les decisions que s'han pres a cada part del projecte.

El periodisme de dades requereix eines complexes, com ara els llenguatges de programació o eines de connexió entre diferents aplicacions com les APIs. En cada cas, hem explicat el funcionament de les eines emprades amb l'ajuda d'imatges quan ha estat possible, i hem procurat facilitar la comprensió dels passatges tècnicament exigents amb la incorporació d'una breu explicació sobre els conceptes clau per poder entendre el codi de programació informàtica que hem fet servir.

Aquest treball serveix per explorar i presentar aquestes eines complexes, i per entendre quina és la manera en la que una disciplina que fa temps que s'incorpora als mitjans internacionals pot trobar un lloc a les redaccions del nostre país.

2 Què és?

2.1.1 Origen històric

És difícil trobar una definició concisa de periodisme de dades. El seu origen es troba en el periodisme de precisió, una disciplina pròpia de la premsa escrita, nascuda als anys 1960 de la mà del reporter americà Philip Meyer. Meyer va ser pioner en l'ús d'ordinadors per generar contingut informatiu durant les revoltes de Detroit de 1967 per *Detroit Free Press*, demostrant que, contràriament al que s'havia pronosticat, els causants dels disturbis no només eren joves sense estudis[1].

Els anys '60 del S. XX van ser testimonis de nombroses innovacions en les rutines periodístiques. Tom Wolfe, Gay Talese i Hunter S. Thompson estaven construint el *New Journalism*, desdibuixant la frontera entre la novel·la i el relat periodístic. El mateix Meyer va descriure aquest moviment com "empènyer el periodisme cap a l'art" en el seu llibre "Precision Journalism: A Reporter's Introduction to Social Science Methods".

Tot i acceptar la validesa del periodisme literari com eina comunicativa, l'autor constata els diferents problemes de producció que es deriven d'una disciplina tan propera a la novel·la, i proposa empènyer el periodisme cap a la ciència[2]. En essència, el periodisme de precisió seria aquell que adopta les tècniques derivades de les ciències socials: partir d'un raonament hipotètic que es posa a prova a través d'eines com l'anàlisi estadística de dades.

2.1.2 El *Big Data*

El naixement d'Internet i la seva massificació van donar lloc als tres fenòmens que han propiciat l'aparició del periodisme de dades. Per un costat, els ordinadors personals s'han imposat com eina de treball fonamental a les redaccions. Per altra banda, ha aparegut l'explosió de dades coneguda com a *Big Data*. Per últim lloc, les millores en hardware són tan ràpides que els ordinadors personals permeten fer anàlisis complexes de grans quantitats d'aquestes dades.

---

[1] Meyer, Philip (8/12/ 2011) Riot theory is relative. *The Guardian.* recuperat des de http://www.theguardian.com/commentisfree/2011/dec/09/riot-theory-relative-detroit-england

[2] Meyer, Philip (2002) - *Precision Journalism: A Reporter's Introduction to Social Science Methods.* Pàg. 4 Oxford: Rowman & Littlefield

Telèfons mòbils, ordinadors, vehicles, objectes equipats amb sensors, ciutats intel·ligents, etc. inunden la xarxa amb dades que queden enregistrades i són disponibles des de qualsevol giny connectat i que poden ser analitzades i tractades des dels ordinadors personals de qualsevol redacció[3].

Sota aquest nou context, el periodisme de precisió s'ha transformat en una nova disciplina, coneguda com a periodisme de dades, que fa servir les eines derivades de les ciències socials aplicades al *Big Data*. Cal tenir en compte també que l'aparició d'Internet ha propiciat una migració de la producció i el consum del periodisme escrit des del paper cap a la pantalla. El periodisme de dades, en tant que disciplina nascuda després de l'aparició d'aquesta migració, incorpora noves maneres de transmetre informació derivades del periodisme en línia com ara apps o gràfics interactius.

2.2 Rutines del periodisme de dades

En les seves formes més elementals, el periodisme de dades combina:
1) El tractament de dades com una font que cal recollir i validar,
2) l'aplicació de tècniques estadístiques per analitzar-la,
3) la confecció de visualitzacions per presentar-la[4].

D'aquestes tres passes es deriva naturalment un llistat de professions involucrades en el procés de crear una peça. En primer lloc, és necessari un periodista amb la capacitat d'extreure el contingut informatiu rellevant de les dades i contextualitzar-lo. Això implica tenir un coneixement profund de l'actualitat que li permeti, un cop tingui les dades processades i convertides en gràfics, descriure-les i detectar quins són els fets que es desprenen d'elles i que tenen rellevància informativa.

En segon lloc, un programador informàtic, que tindrà diferents funcions: en primer lloc, ajudar amb el procés de recollir dades. Com veurem més endavant, existeixen moltes fonts diferents de dades (taules estadístiques, xarxes socials, informes, etc.), codificades en un formats diferent (PDF, CSV, XLS, etc.) i que necessiten mètodes específics per ser recollides i tractades. En segon lloc, el programador pot ajudar en el tractament estadístic, ja que quan les bases de dades són molt grans, és habitual fer servir programació

[3] Shaw, Jonathan (2014). Why big data is a big deal. Harvard Magazine, des de http://harvardmagazine.com/2014/03/why-big-data-is-a-big-deal

[4] Benjamin Howard, Alexander (2014) *The art and science of Data-driven journalism*. Nova York: Columbia School of Journalism

informàtica per obtenir resultats. Finalment, el programador porta a terme la part tècnica darrere del disseny gràfic per fer funcional les visualitzacions confeccionades pel dissenyador.

El dissenyador és qui ha d'entendre el contingut de la peça i trobar la manera més adient de comunicar-lo de manera gràfica. Existeixen diferents solucions per dissenyar una peça: es poden fer servir eines que automatitzen el procés de disseny, però això implica tenir molt poc control sobre els aspectes visuals. La manera ideal és treballar amb un dissenyador i un informàtic, de manera que el primer no treballa constret per limitacions tecnològiques perquè el segon és capaç de materialitzar la seva visió.

2.3 El nostre entorn

Un equip com el descrit anteriorment podria produir peces per qualsevol mitjà de comunicació digital. Des d'un punt de vista empresarial, existeixen dues maneres per incloure'l: incorporar-lo a la redacció com una secció més, de la mateixa manera que existeixen les seccions d'internacional, economia, o continguts digitals; o externalitzar aquesta tasca, comprant peces en exclusivitat a empreses que es dediquin a vendre aquest producte.

Donant un cop d'ull al mercat de mitjans de comunicació espanyols, el primer que detectem és que no existeixen empreses del segon tipus. Les grans capçaleres tradicionals, però, sí que estan començant a incorporar equips similars, alguns d'ells més complexes, a les seves redaccions. També existeix un cas excepcional de productors de periodisme de dades, l'organització Civio.

2.3.1 Civio

Es tracta d'una organització sense ànim de lucre en favor de l'administració transparent que produeix exercicis de periodisme de dades. No venen a altres mitjans de comunicació, i no tenen ànim de lucre. Tenen especial interès pel nostre treball, ja que publiquen els seus resultats financers anuals al seu web.

Per entendre quin tipus d'entitat es tracta, hem consultat la secció de consultes jurídiques freqüents al web del col·legi d'advocats de Madrid[5]. Hi hem descobert que una empresa

[5] Ilustre Colegio de Abogados de Madrid, consultas jurídicas frecuentes, a http://crsa.icam.es/web3/cache/CRSA_asesora.html

sense ànim de lucre pot realitzar tota classe d'activitats econòmiques que estiguin relacionades amb els seus fins (en el cas de Civio, potenciar la democràcia oberta) i han de reinvertir-hi els beneficis econòmics. Al seu web trobem que donen serveis d' assessoria en transparència de dades i formació tant a empreses com institucions, i que no reben ingressos per publicitat.

Seguidament, hem estudiat els resultats financers de 2015[6]. Comprovem que en aquest exercici han obtingut 227.000 € d'ingressos de l'activitat pròpia, distribuïts en 19.000 per donacions, 72.000 per prestació de serveis i 135.000 en suport institucional. En les despeses, destaquen 139.000 € en salaris de sis treballadors a temps. El resultat total després d'impostos és de 49.600 €.

Civio és un model únic, que destaca per l'evolució de les seves xifres. En tres anys de vida, la xifra d'ingressos ha escalat des dels 94.000 € del primer exercici fins als 227.000 de l'últim. La plantilla s'ha ampliat de dos treballadors inicials a sis treballadors a temps complet. La major part d'aquests ingressos s'ha donat gràcies a les ajudes rebudes al capítol de suport institucionals.

2.3.2 Mitjans amb secció de dades

Elconfidencial.com: El Confidencial (noteu que no es tracta del mateix mitjà que el confidencial digital) té la secció més veterana (2013) de periodisme de dades dels diaris nacionals, i la fa amb set periodistes de la casa. El Confidencial publica els integrants de la seva plantilla al web http://www.elconfidencial.com/somos/, i hem apreciat que existeixen diverses seccions convencionals (actualitat, esports, cultura, etc.) i una secció anomenada *Laboratorio: proyectos y área visual*, formada per 12 persones, entre les quals una unitat de periodisme de dades formada per 3 treballadors.

Eldiario.es: aquest mitjà no té una secció activa de periodisme de dades, però incorpora la disciplina com a eina periodística a moltes de les seves peces des de l'any 2014. El gruix d'aquests articles estan signats per Raúl Sánchez, periodista de dades, i Belén Picazo, dissenyadora.

El Mundo: La seva secció de dades s'anomena El Mundo Data, i publiquen amb una freqüència d'una peça mensual molt complexa i profunda, amb firmes de 5 professionals (Marta Ley, Paula Guisado, Hugo Garrido, Javier J. Barriocanal y Pablo Medina).

[6] http://www.civio.es/dev/wp-content/uploads/2016/04/01_Cuentas-2015-INFORME-ANUAL-copy.xlsx

El Español: Aquest mitjà existeix des d'octubre de 2015, i va començar fent una aposta decidida pel periodisme de dades: publicava una mitjana d'una peça diària en diferents seccions temàtiques (principalment política, esports i internacional) des de novembre fins febrer. A partir de llavors, a causa d'una reestructuració interna que va portar a una reducció de plantilla, la freqüència de les peces s'ha reduït fins a una mitjana de 3 al mes.

La Vanguardia: la versió digital d'aquest diari té una secció de periodisme de dades anomenada Vangdata, subvencionada pel Ministeri d'Educació, Cultura i Esports. Està dirigida per Laura Aragó, i produeix una mitjana de tres peces cada dues setmanes. Cal tenir en compte que no tot el que publiquen són exercicis de periodisme de dades amb l'estructura ortodoxa que hem presentat abans. Algunes peces tenen les dades com a protagonista, i incorporen imatges fixes de gràfics, però no són interactives i no hi ha una anàlisi de dades dut a terme a la redacció. En alguns casos, són peces elaborades a altres diaris que s'han traduït i adaptat[7].

Altres: Elperiodico.es: la versió digital d'el Periódico de Catalunya disposa d'una secció de dades que no s'actualitza amb freqüència i que no està activa en el últims mesos. El País ha etiquetat com a "periodisme de dades" un total de 9 peces entre 2012 i 2016. A més, no es tractava d'exercicis de periodisme de dades sinó notícies convencionals que tractaven el tema.

2.3 El nostre exemple

Aquest treball consisteix en un exercici de periodisme de dades del que després farem una breu anàlisi de les rutines productives per estudiar la viabilitat de la incorporació d'un periodista de dades a la redacció d'un mitjà. Al punt següent diseccionem totes les parts en què consisteix un reportatge de dades, i a l'últim apartat, per cada pas fem un estudi de tots els recursos productius necessaris per dur-lo a terme. A través d'aquest exercici, es posen de manifest les rutines productives i el seu encaix en un mitjà de comunicació.

L'exemple que presentem és el d'una anàlisi de lèxic polític a xarxes socials, concretament a Twitter, en el context de les eleccions espanyoles del 20 de desembre de 2015, que donarà peu a una breu notícia que incorpora elements interactius, tots derivats de l'anàlisi de les dades que hem recollit, analitzat i visualitzat.

[7] En el mundo, ya hay más obesos que flacos (03/04/2016) *La Vanguardia,* a http://www.lavanguardia.com/vangdata/20160403/40847295800/obesidad-grafico-mapa-mundo-raking-paises.html

3 Creació de la peça

3.1 La idea

L'objectiu d'aquest exercici és analitzar el cos de paraules que han fet servir els partits polítics durant les eleccions per confeccionar una peça periodística. Per arribar fins aquí, cal dur a terme les següents fases:

1) Elecció de la plataforma: Twitter
2) Delimitació de la mostra:
    1) De quins usuaris recollirem les dades?
    2) Com filtrem els missatges?
3) Extracció de dades de Twitter
4) Anàlisi de les dades
5) Visualització
6) Construcció de la peça periodística

3.1.1 Elecció de la plataforma

En el nostre cas, hem decidit fer servir la plataforma Twitter per dos motius bàsics:

1) Té una gran abundància de missatges per part dels partits polítics, que tenen compte oficial i que l'han fet servir abundantment durant la campanya.
2) Facilitat d'extracció: Twitter permet connectar amb altres aplicacions per tal d'extreure bases de dades filtrades de la manera que nosaltres volguem, amb dades de contingut, nom d'usuari, dia i hora del tuit i geolocalització, entre altres.

3.2 Recollida de dades

3.2.1 Delimitació de la mostra

Hem confeccionat una gran base de dades amb contingut extret de Twitter. Recull els missatges dels quatre grans partits polítics que es presentaven a les eleccions i dels seus quatre líders, que continguessin paraules d'un petit diccionari que termes que fan referència al procés d'independència a Catalunya. Amb aquesta, pretenem després fer una anàlisis de freqüència de paraules per esbrinar quin tipus de paraules feien servir els polítics a l'hora de parlar del tema de l'autodeterminació a Catalunya.

Piulades de polítics	
Temps	Duració íntegra de la campanya electoral (des del 4 de desembre fins al 19 de desembre de 2015)
Usuaris	8 - Els quatre comptes oficials dels partits polítics i els quatre comptes oficials dels seus líders: Podemos (@ahorapodemos - @Pablo_Iglesias_) Partit Popular (@ppopular @mariajorajoy) Ciutadans (@ciudadanoscs - @albert_rivera) PSOE (@PSOE - @sanchezcastejon)
Paraules clau	Catalunya - Cataluña - Referendum - Autodeterminación - Autodeterminació - Catalan - Català
Etiquetes	Totes
Total	**13.520** piulades

3.2.2 Eines emprades

Les dades es poden obtenir de moltes maneres, però la més habitual és a través de *datasets* que ja estiguin publicats. En serien un bon exemple les taules estadístiques que publiquen els òrgans oficials com l'IDESCAT. Però no sempre existeix una entitat que hagi recollit, classificat i formatat les dades de la manera que nosaltres volem. [8]

En aquest cas, cal que el periodista confeccioni la seva pròpia base de dades. L'objectiu és tenir un full de càlcul amb la informació que nosaltres requerim, ja que aquest és el format que permet treballar amb les dades i analitzar-les. Els periodistes de dades intenten transformar informació d'una gran quantitat de diferents plataformes en línia en bases de dades, i aquest procés rep el nom de *scraping*. Es pot aplicar aquesta tècnica mitjançant diferents eines a:

- Pàgines web. Per posar un exemple, la pàgina NEWS (www-news.iaea.org (sic)) és un blog que recull els accidents i anomalies conegudes a les centrals nuclears d'arreu del món. Cada entrada del blog correspon a un accident, i ve acompanyada de molta informació. A

[8] Gray, Jonathan; Chambers, Lucy; Bounegru, Liliana (2012) *The Data Journalism Handbook,* Sebastopol(EEUU): O'Reilly Media

través d'una eina de *scraping* es pot crear una base de dades on cada accident estigui classificat en funció del nom de la central, el país, el dia i l'hora, el tipus d'anomalia, la cobertura, etc.

- PDFs: moltes organitzacions i entitats posen a disposició del públic dades (resultats financers, llistes de tot tipus, etc) en format PDF. Aquest format està pensat per la impressió, i permet escalar les tipografies sense perdre resolució, ja que estan representades com línies rectes i corbes representades en un determinat punt - un procés que es coneix com *vectorització*. El costat negatiu, però, és que el PDF ignora l'estructura i jerarquia de les dades que s'hi representen. Si hi trobem una taula estadística, el programa no sap interpretar quina és la columna que dóna nom als diferents valors - de fet, ni tan sols és capaç d'entendre una taula de valors. Amb eines *d'scraping* és possible traduir aquestes dades de manera automatitzada a un format de full de càlcul.

3.3.3 APIs

Es tracta d'una altra manera d'obtenir dades a temps real d'un servei web. API són les sigles angleses d'Interfície de Programació d'Aplicacions, i la definició tècnica és "un conjunt d'indicacions, quant a funcions i procediments, ofert per una biblioteca informàtica o programoteca per ser utilitzat per un altre programa per interaccionar amb el programa en qüestió"[9]. Dit d'una altra manera, és allò que permet dos serveis diferents comunicar-se entre si.

Per exemple, la coneguda aplicació de *streaming* musical Spotify és compatible amb aquest sistema[10]. Mitjançant APIs, els programadors que tinguin webs sobre música poden fer-les entendre amb Spotify, per permetre als seus visitants a crear llistes de reproducció compartides, per exemple.

O per fer coses molt més sofisticades. "30s Drum Machine[11]" és una caixa de ritmes virtual que fa servir com *samples* els 30 primers segons de cançons d'Spotify. Aquestes estan classificades amb una gran quantitat de dades: nom de l'artista, lloc de procedència, nombre de reproduccions, però també estil musical i BPM (pulsacions per minut, el concepte musical que mesura la velocitat del ritme d'una cançó). L'API posa en

[9] Interfície de programació d'aplicacions. *Wikipedia* , a https://ca.wikipedia.org/wiki/Interf%C3%ADcie_de_programació_d%27aplicacions#Alguns_exemples_d.27API

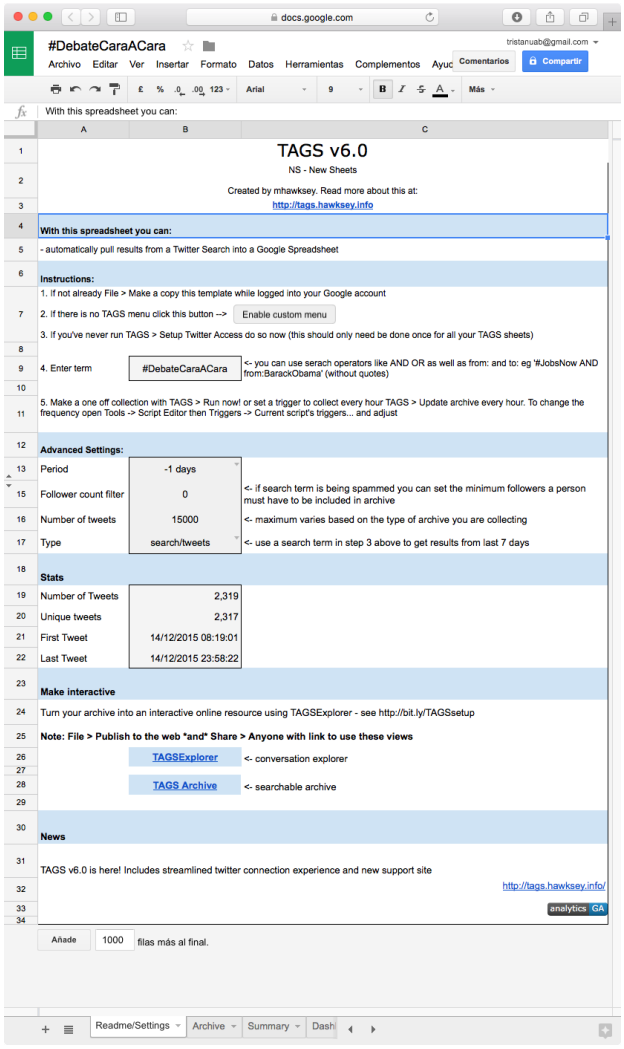
[10] Web API Code Examples & Libraries. *Spotify Developer,* consultat el 2016 a https://developer.spotify.com/web-api/code-examples/

[11] "30s Drum Machine" http://lab.possan.se/druuum/

contacte el web amb Spotify i s'encarrega de filtrar i descarregar cançons amb el mateix BPM, de tal manera que els ritmes quadrin; i d'estils musicals amb gran presència de percussió com ara música electrònica o rock. La interfície és la d'una petita taula de mescles, que permetre l'usuari fer mescles entre moltes cançons diferents.

3.3.4 TAGS i Google Sheets

Aplicades a les xarxes socials, les APIs permeten als periodistes obtenir dades en temps real de l'estat del debat públic d'un tema concret. En el nostre cas, hem fet servir una API per connectar Twitter a un programa de base de dades. La que hem fet servir rep el nom de TAGS. Es tracta d'una eina gratuïta desenvolupada per Martin Hawskey des de 2010, que funciona amb Google Sheets, el programa de full de càlcul al núvol de la companyia californiana.



TAGS es carrega com un full de càlcul de *Sheets*, des d'on cal configurar l'aplicació amb les credencials de Twitter perquè hi pugui tenir accés, procés que cal repetir a la inversa des de la xarxa social. Un cop fet això, els dos programes s'entenen i pot començar el flux de dades entre ells. A la primera pantalla, podem veure les opcions de *scraping* que permet TAGS: podem configurar una paraula o un usuari a seguir al punt 4 – en el nostre exemple, hem fet una captura del document que filtrava les piulades del debat del 14 de desembre amb l'etiqueta #debatecaraacara. També permet recuperar les piulades dels anteriors 7 dies i limitar aquest període, permet filtrar les piulades segons un nombre mínim de seguidors dels seus autors per assegurar-se obtenir resultats de comptes amb certa rellevància. Permet fer cerques de piulades, de llistes i favorits o de *timelines* concrets. A la captura de la dreta, veiem els resultats de l'aplicació de l'API.

TAGS recull la següent informació de cada piulada:

- ID únic (Twitter assigna un ID únic, un número de 18 xifres, a cada participació),	- nom de l'usuari a qui replica en cas de ser una rèplica,
- dia,	- sistema operatiu del dispositiu (Android, iOs, Windows, MacOS, etc),
- hora,	- nombre de seguidors de l'usuari
- nom de l'usuari,	- nombre d'amics de l'usuari
- contingut de la piulada,	- enllaç a la foto de perfil
- geolocalització,	- enllaç permanent a la piulada
- idioma en què l'usuari té configurada l'aplicació,	

El resultat d'aquest procés és un full de càlcul de *Spreasheets* on es poden veure totes les dades. El funcionament és en viu i en dos sentits: recull tots aquells tuits que compleixen els requisits marcats fins al període de temps que hem marcat, i, a partir d'aquí, un cop per hora recull en directe aquells que hagin aparegut. Durant aquest procés, podem interrompre la màquina i tornar-la a posar en marxa, i el programa s'encarrega de recuperar els missatges que s'han emès mentre estava desconnectat. És una bona pràctica dur a terme còpies de seguretat periòdiques, i assegurar-se que el programa segueix funcionant i que no hi ha problemes de connexió amb Twitter.

Cal tenir en compte dos factors a l'hora de planificar aquest procés. El primer és l'extensió de la mostra recollida: en funció dels paràmetres que fem servir per filtrar la cerca, és possible que la quantitat de missatges enregistrats sigui molt alta. L'API ens demana marcar un límit de piulades, i el valor que proposa per defecte és 15.000. A partir d'aquesta xifra, és possible que la base de dades es torni massa gran i el nostre equip comenci a tenir problemes per manejar-la. En casos en què es vulgui obtenir una quantitat molt alta de piulades, és important anar desant el full de càlcul periòdicament i netejant el document de *Spreadsheets*.

El segon factor són les limitacions de Twitter a l'hora de fer servir APIS. Aquest enorme volum de missatges implica una gran càrrega pels servidors de la xarxa social, i per aquest motiu les consultes es limiten de diferents maneres[12]. La més destacada és que, via API, Twitter només retorna les piulades de menys d'una setmana d'antiguitat[13]. Això condiciona la nostra rutina professional, ja que ens obliga a determinar el tema del què tractarem amb no més de 7 dies de distància sobre el fet.

3.3.5 Format de les dades: CSV

Un cop ha finalitzat la recollida de dades, cal descarregar el full de càlcul amb les piulades al disc dur per començar a analitzar-lo. Recordem que un full de càlcul és un programa informàtic que permet manipular dades alfanumèriques (que contenen xifres i lletres) disposades en forma de taules compreses per files i columnes, i les seves interseccions, anomenades cel·les[14]. Els fulls de càlcul poden treballar amb fitxers de diferents extensions, cada un amb diferents característiques.

Per exemple, XML és un tipus de fitxer complex que permet etiquetar i afegir notes a les cel·les però és molt exigent amb el maquinari, JSON és més simple i més lleuger, els fulls de càlcul com Spreadsheets o XLS permeten aplicar fàcilment fórmules amb programes com Excel[15]. En el nostre cas, farem servir un dels formats més simples: CSV. Aquest tipus d'arxiu consisteix en una simple successió de línies formades per cel·les separades per comes, com indica el seu nom (Comma Separated Values, Valors Separats per Comes

---

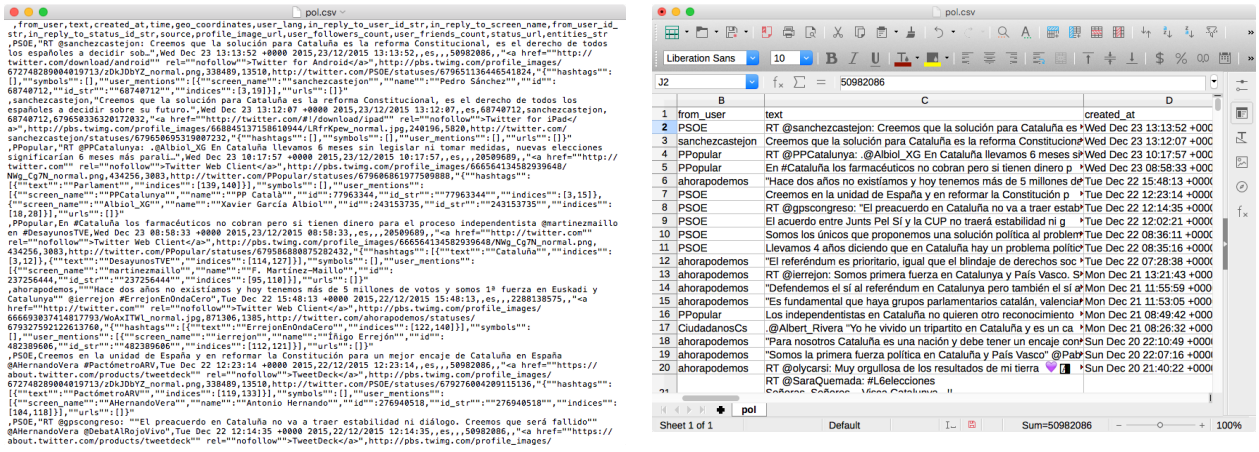
[12] Things every developer sould know, *Twitter / Developers*, consultat el 2016 a https://dev.twitter.com/overview/general/things-every-developer-should-know

[13] Font: Twitter.com, pàgina per desenvolupadors, secció APIS, capítol "The search API" a https://dev.twitter.com/rest/public/search

[14] Fulls de càlcul. *Wikipedia.* a https://ca.wikipedia.org/wiki/Full_de_càlcul

[15] Ventouris, Anastasios (2013) A Python guide for open data file formats. *Open Knowledge Foundation Labs,* consultat el 2016 des de http://okfnlabs.org/blog/2013/10/17/python-guide-for-file-formats.html

en anglés). El seu principal avantatge és que és extremadament lleuger, de tal manera que una base de dades amb 13.520 línies amb 17 valors cada una, com la nostra, es pot manipular en segons. Els llenguatges de programació que es fan servir per analitzar dades, com ara R o Python, estan preparats per treballar de forma nativa amb CSV, i això també és un factor clau a l'hora de decantar-se per aquest format. Finalment, Spreadsheets permet exportar automàticament qualsevol taula en CSV, per tant l'elecció d'aquest format no ens comporta cap despesa de recursos addicional.



El nostre arxiu CSV, a l'esquerra contingut cru, i a la dreta, interpretat per un programa de full de càlcul, en aquest cas LibreOffice

A partir d'aquí, és possible que alguns arxius s'hagin de netejar. Això succeeix quan el format d'algun dels valors no és compatible amb el programa que fem servir per analitzar-lo: un exemple arquetípic són les dates en format americà, on "22/12/2015", estaria escrit com "12/22/2015". Aquests errors poden distorsionar severament els resultats de les anàlisis de dades[16]. En aquest sentit, ens cal tenir dues eines fonamentals. En primer lloc, cal un full de càlculs, que permeti les operacions simples. Els programes habituals són Excel de Microsoft, Numbers d'Apple, OpenOffice d'Apache, i LibreOffice, una versió molt lleugera d'aquest últim. LibreOffice ens sembla l'opció més adecuada, per la seva lleugeresa i el fet que és gratuït. També cal un programa per manipular grans bases de dades de manera ràpida, i en aquest sentit encara no hi ha un substitut per Google Open Refine. En el nostre cas, les manipulacions necessàries seran mínimes, i les podem dur a terme amb el llenguatge de programació que veurem a continuació.

[16] Hellerstein, Joseph M. (2008) *Quantitative Data Cleaning for Large Databases*, pàgina 1 Boston: UC Berkeley, consultat el 2016 des de http://db.cs.berkeley.edu/jmh/papers/cleaning-unece.pdf

3.4 Tractament de les dades

3.4.1 Elecció de l'eina: Python

Existeixen molts camins per arribar des d'una base de dades a una peça periodística, es poden fer servir eines com Google Fusion Tables, Excel, o directament eines de visualització com Tableau o DataMarket [17]. Però la forma més profunda de tractar amb dades és a través d'un llenguatge de programació [18], d'entre els quals destaquen R i Python.

Per una banda, R està pensat específicament per anàlisi de dades, i es fa sevir principalment a l'àmbit acadèmic i en recerca. Python, per altra banda, és més polivalent i estès al món empresarial, però destaca per ser molt potent en tractament de dades i especialment per ser fàcil de llegir, en el sentit que la seva sintaxi és més intuïtiva. Els dos són gratuïts i tenen moltes llibreries que visualitzacions de dades que fer tota mena de gràfics interactius, tot i que la de R és més amplia. A l'hora de treballar amb el codi, Python permet fer servir entorns de desenvolupament que integren aquestes visualitzacions, i permeten acolorir la sintaxi i dividir el codi en diferents seccions. Aquestes eines i la sintaxi intuïtiva fan que es consideri que Python és un bon llenguatge per començar a aprendre a programar, i aquest és el motiu pel qual hem optat per ell [19].

3.4.2 Eines necessàries:

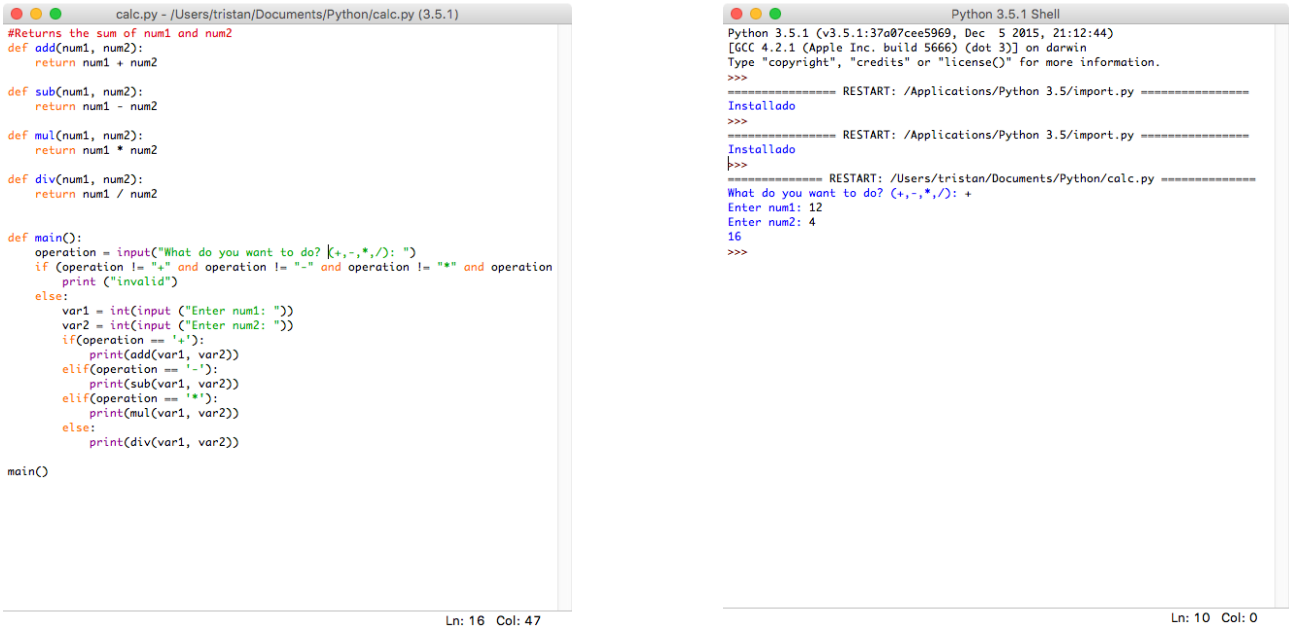
1. El llenguatge. Python és un llenguatge gratuït multiplataforma, que ve integrat a l'entorn Macintosh, la qual cosa significa que ve instal·lat per defecte a tots els Macs. Es pot descarregar la versió més actual des de https://www.python.org. Per defecte, Python permet programar codi en una finestra que rep el nom d'IDLE, i visualitzar-los en una finestra a part, coneguda com a shell.

---

[17] Remington, Alex (8/5/2016) Understanding data journalism: Overview of resources, tools and topics. *Journalists resouce,* consultat el 2016 a http://journalistsresource.org/tip-sheets/research/understanding-data-journalism-overview-tools-topics

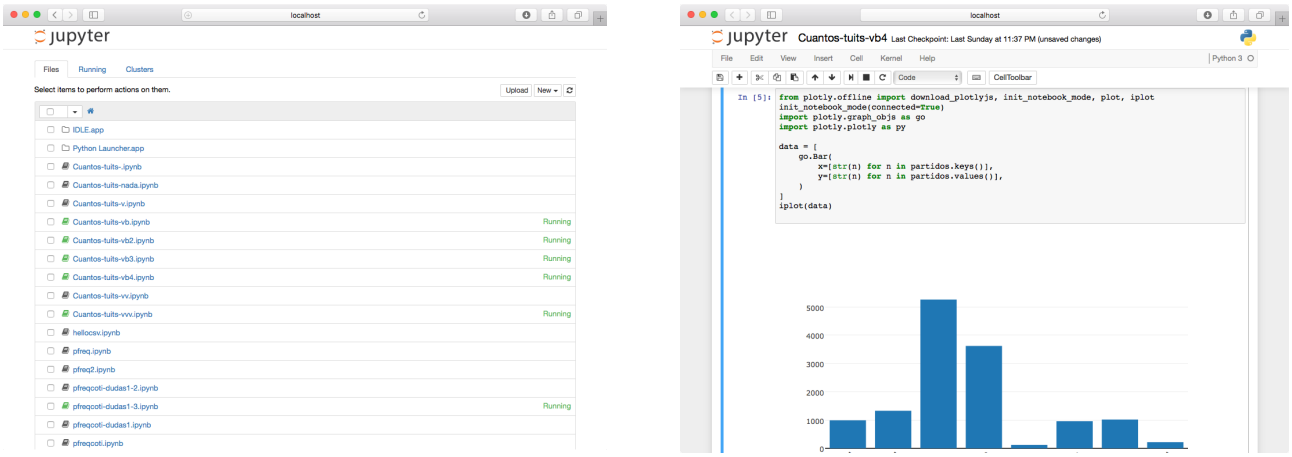
[18] Dyer, Zach (17/05/2015) Journalists' beginner guide to coding. *Journalism in the Americas*, consultat el 2016 a https://knightcenter.utexas.edu/blog/00-13913-journalists'-beginner-guide-coding

[19] The Datacamp Team (12/05/2015) Choosing R or Python for data analysis? An infographic. *Datacamp,* consultat el 2016 a https://www.datacamp.com/community/tutorials/r-or-python-for-data-analysis



Les dues finestres de Python

2. Jupyter. Es tracta d'un entorn de desenvolupament. Això és una eina informàtica per al desenvolupament que permet escriure codi de manera còmoda i ràpida, que agrupa diferents funcions en un sol programa, habitualment: editor de codi, compilador, depurador i un programa de disseny d'interfície gràfica[20]. Jupyter és gratuït, es pot descarregar a http://jupyter.org i permet separar el codi per seccions, escriure notes, acolorir la sintaxi, guardar i compartir codi al núvol i integrar visualitzacions. Funciona amb qualsevol navegador modern.



Jupyter, el navegador i l'editor de codi amb gràfics integrats

3. Mòduls. Es tracta de fragments de codi de Python que contenen definicions que permeten dur a terme funcions molt concretes, i que podem carregar al nostre programa

[20] Margaret Rouse (febrer de 2007). Integrated Development Environment (IDE). *TechTarget,* consultat a http://searchsoftwarequality.techtarget.com/definition/integrated-development-environment

[21]. Explicat d'una altra forma, són complements que descarreguem d'internet i fem servir al nostre programa per realitzar tasques molt específiques com comptar les paraules d'una llista o crear gràfics. Farem servir els següents 6 mòduls, tots ells gratuïts:

- CSV: permet obrir i interpretar bases de dades CSV.
- Counter: permet comptar els elements d'una llista
- re: permet identificar el conjunt de lletres separats per espai com una paraula (si no el féssim servir, només podríem obtenir la freqüència d'ús de les 24 lletres de l'abecedari)
- string: el farem servir per transformar totes les paraules en minúscules, per evitar duplicitat de resultats
- stop_words: permet creuar llistes amb diccionaris de paraules buides
- plot.ly: permet elaborar i configurar molts tipus de gràfics interactius

3.4.3: El codi

1 Presentació

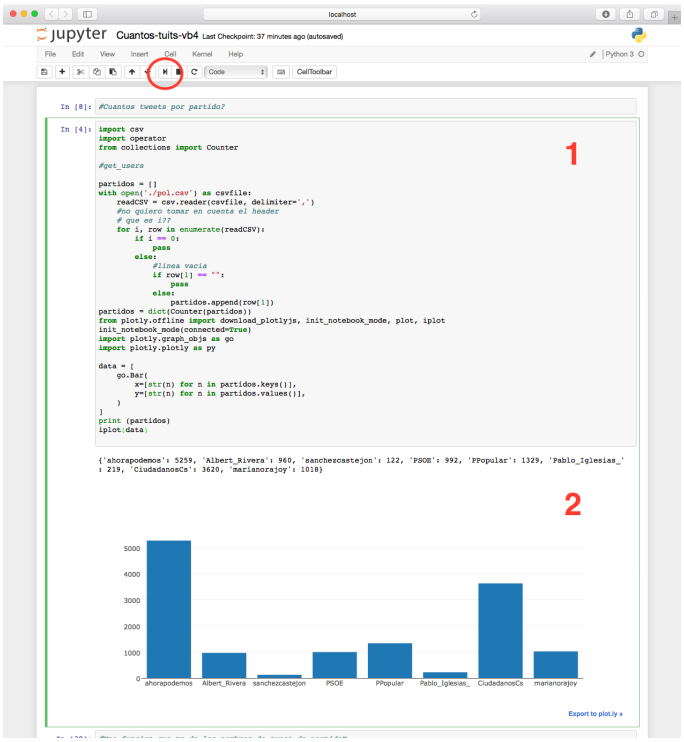
El nostre objectiu és obtenir diferents gràfics que ens permetin entendre quines han estat les paraules que més han fet servir els partits polítics a Twitter durant la campanya, partint de la nostra base de dades. Per arribar fins aquí, hem de fer servir diferents porcions de codi que permeten ordenar, aïllar, distribuir i relacionar les dades. El nostre codi és una mena d'arbre, que té les següents seccions, que tenen com a resultat la creació de variables que contenen informació ordenada i que es poden relacionar amb altres:

1) La llista de partits i polítics, i el nombre de tuits per cada un
2) Una funció que permeti veure tots els tuits d'un usuari, és a dir, d'un partit o polític
3) La llista de les paraules (tags) més usades en general, que anomenarem paraules clau
4) La llista de les paraules clau filtrades segons el partit o líder que les ha emès

Explicarem les dues primeres i l'última en detall, ja que per la tercera hem fet servir el mateix codi que es deriva de les dues primeres. Primer, però, ens cal fer una breu explicació de les nocions bàsiques de python.

[21] Documentació en línia de Python, capítol 6: Modules, a https://docs.python.org/3/tutorial/modules.html

Executar el codi:



En aquesta imatge de Jupyter, l'entorn gràfic on escriurem el codi, es poden apreciar els dos elements fonamentals. A la part grisa (1) és on escriurem el codi. Un cop haguem acabat una porció de codi, premem el botó encerclat en vermell a la barra d'eines de la part superior. Això executarà el codi, i en mostrarà el resultat a la part blanca inferior (2), que anomenarem consola. Aquesta és l'acció a la que ens referim quan diem "processar el codi".

Parts de Jupyter

2 Conceptes fonamentals per entendre aquestes línies de codi:

Sintaxi bàsica:

`#` al principi d'una línia la desactiva. Es fa servir per reservar codi o deixar notes.

`+ - * %` Signes per les operacions matemàtiques de sumar, restar, multiplicar i dividir

`=` el valor que té a la seva dreta s'assigna a la variable que te a l'esquerra

`in, not in` avaluar si una condició es troba (o no) a la seqüència especificada

1) Variables

```
var1 = 100
```

Una variable és un valor o conjunt de valors (en aquest cas, el numero 100) als que assignem una etiqueta – en aquest cas, "var1". Això ens permet aplicar tot tipus d'operacions a una sèrie tancada d'elements, referint-nos al nom "var1": sumar-los o restar-los, comptar-los, ordenar-los, etc. Aquesta acció sol rebre el nom de "cridar *(to call)* una variable". En el nostre treball, en farem servir de 4 tipus: numèriques (com a l'exemple), strings (contenen lletres i números, i no són calculables, van entre cometes), i llistes i diccionaris, que veurem a continuació.

2) Llistes (i elements indexats)

`llista1 = [12,8,6,4,2]`

Aquest codi crea una llista anomenada “llista1” que conté els valors 12, 8, 6, 4 i 2. Una llista és un conjunt d’elements d’indexats, separats per comes. “Indexat” vol dir que cada un dels valors té un número assignat segons la seva posició, començant per 0. A l’exemple, l’índex 0 correspon al número 12 perquè és el primer de la llista, l’índex 1 correspon a 8 que és el segon, l’índex 2 correspon al número 6 i així successivament.

Es pot crear una variable buida per omplir-la més endavant, deixant els claudàtors buits: `llistabuida = []` Extend i append:

3) Diccionaris

dic1 = {‘dilluns’: 1, ‘dimarts’:2, ‘dimecres’:3}

Aquesta sintaxi crea un diccionari. Un diccionari és una llista de parelles d’elements, separades per comes. Cal pensar en els diccionaris de llengua, que són enormes col·leccions de parelles formades per un terme i la seva definició. El primer element de la parella, el terme, s’anomena “key”; i el segon, la definició, s’anomena “value”. Igual que les llistes, el seu contingut està indexat, es poden crear diccionaris buits i es poden “cridar”.

4) Funció (function)

```
def fun(paràmetres):
    contingut
```

Aquesta sintaxi d’exemple crea una funció anomenada “fun”. Una funció és un bloc de codi organitzat i reutilitzable, que es fa servir per dur a terme una acció una vegada. El contingut de la funció aniria tabulat a les següents línies. L’objectiu és “cridar” més endavant tot aquest codi amb una sola paraula (a l’exemple, “fun”), com es pot fer amb les llistes i els diccionaris.

A part de les funcions definides per l’usuari, Python disposa de 76 funcions preinstal·lades, que sempre estan disponibles. Les que farem servir en el nostre codi són:

`print()` mostra el contingut de qualsevol objecte (llista, diccionari, etc.) a la consola.

`enumerate()` analitza una llista i retorna un diccionar amb un numero assignat a cada un dels seus elements.

`count()` analitza una llista i retorna el número total d’elements

`open()` obre un arxiu (en aquest cas, el nostre CSV).

`dict()` manera alternativa de crear un diccionari

`str()` permet fer modificacions a caràcters alfanumèrics (que contenen xifres i lletres), com passar de majúscules a minúscules

`return()` obté (i mostra a la consola) el resultat dels càlculs d'una funció.

5) Condicionants *if, then, else*

Aquestes sentencies són condicionants que ens permeten fer avançar el càlcul en dues possibles direccions en funció de determinats parametres. *If*, equivalent anglès de la conjunció condicional "si", marca una condició que cal complir i el codi subseqüent. *Else,* equivalent a "sinó" o "en altre cas", conté el codi a executar si no es compleix aquesta condició.

6) Bucle *For*

Aquesta sentència permet crear un bucle o cicle: repeteix una ordre per cada un dels elements d'una cadena. Explicat d'una altra manera, permet repetir un fragment de codi un cert nombre de vegades.

Codi 1 Llista de partits i nombre de piulades per partit:

```
import csv
from collections import Counter

partidos = []
with open('./pol.csv') as csvfile:
    readCSV = csv.reader(csvfile, delimiter=',')
    for i, row in enumerate(readCSV):
        if i == 0:
            pass
        else:
            #linea vacia
            if row[1] == "":
                pass
            else:
                partidos.append(row[1])
partidos = dict(Counter(partidos))
```

Primera part del codi

Al nostre arxiu CSV tenim moltes columnes amb diferents elements: el nom de l'usuari que ha escrit la piulada, el seu contingut, l'hora, la data, la geolocalització, l'idioma, etc. Necessitem assignar aquestes dades a funcions i diccionaris, de tal manera que puguem dur a terme operacions matemàtiques sobre elles més endavant. En primer lloc, volem obtenir una llista d'usuaris. Aprofitarem per comptar quantes vegades apareixen a la columna, i això ens indicarà quantes piulades han fet.

El primer que hem fet és importar els mòduls CSV i Counter, com veiem a les primeres dues línies. El primer ens permet obrir l'arxiu CSV, i el segon ens permet comptar els elements d'una llista. Després, amb `partidos = []`, declarem la variable "partidos". De moment està buida, ja que hi inclourem elements més endavant. Amb la sentència `with open`, que permet obrir un arxiu i tancar-lo després d'executar el següent codi, obrim el nostre CSV i n'extraiem informació.

Neteja de dades: en el nostre cas, trobem que TAGs ha generat algunes línies o cel·les buides que poden contaminar el resultat. Hem optat per fer servir Python per solucionar-ho, a través dels condicionants *else/if*. En primer lloc, cal eliminar l'encapçalament o *header*. El *header* és la primera fila de cada full de càlcul que indica quin és el contingut de la columna que té a sota. Amb el codi `for i, row in enumerate(readCSV):` declarem un bucle `for`, que actuarà sobre `i` i `row` (línies i columnes), tants cops com cel.les hi hagi a la llista `(enumerate(readCSV))`. A partir d'aquí, creem dos camins: quan el processador trobi la primera columna que té l'índex 0 `(if i == 0)`, el dirigim a no fer res i seguir amb la següent instrucció `(pass)`, això ignora el *header*. En altre cas `(else:)`, si es trobés una

línia buida (`if row[1] == “”:`), el tornem a dirigir a la següent instrucció. En aquest punt, el processador ha d'estar tractant les cel·les que no són ni l'encapçalament ni cel·les buides, i per tant només actua sobre dades correctes. Podem procedir a crear la llista.

La columna amb l'index 1 és la que conté el nom de l'usuari que ha emès el tuit. A dins el codi, ens referirem a ella com `row[1]`. El codi `partidos.append(row[1])` afegirà a la llista cada paraula que hi trobi. Aquesta contindrà 13.520 índexs: cada cop que hi aparegui un nom d'usuari, voldrà dir que aquest ha emès una piulada. Per tant, només ens queda comptar cada cop que un usuari hi apareix. Ho incorporarem a un diccionari: recordem que un diccionari és una llista de parelles d'elements. Per fer-ho, utilitzem el codi `partidos = dict(Counter(partidos))`. Finalment, el mostrem a la consola amb la instrucció `print()`. Aquest serà el resultat a la consola quan executem:

```
{'PPopular': 1329, 'ahorapodemos': 5259, 'CiudadanosCs': 3620, 'sanchezcastejon': 122,
'Pablo_Iglesias_': 219, 'marianorajoy': 1018, 'PSOE': 992, 'from_user': 1,
'Albert_Rivera': 960}
```

Doble funció del diccionari: recordem que als diccionaris, cada primer element d'una parella és una clau o *key*, i cada segon element és un valor o *value*. En aquest cas, els noms de partits són les keys i el número de piulades, les values. Per tant, si cridem només les keys del diccionari amb el codi `partidos.keys()` (aplicable a qualsevol diccionari), obtindrem un diccionari amb els 8 noms d'usuaris. El convertim en una llista (que contenen elements únics, en lloc de parelles) amb el codi `mis_partidos = partidos.keys()`.

Codi 2 Crear una llista de les paraules més usades

En el cas anterior, hem treballat sobre la columna que contenia el nom d'usuaris, row[1]. Ara, treballarem sobre la que conté les piulades: row[2]. En aquest cas, volem esbrinar amb quanta freqüència s'ha fet servir cada paraula. Després, ordenarem el conjunt en funció de la freqüència i mostrarem només els 100 primers elements. D'aquesta manera, sabrem quines són les 100 paraules més usades.

```
tags = []
with open('./pol.csv') as csvfile:
    readCSV = csv.reader(csvfile, delimiter=',')
    for i, row in enumerate(readCSV):
        if i == 0:
            pass
        else:
            if row[1] == "":
                pass
            else:
                tweet = row[2]
                lista_tags = get_tag_list(tweet)
                tags.extend(lista_tags)
print(len(tags), "tags usados en este fichero")
```

Codi per aplicar diccionaris abans de crear una llista

El primer problema que ens trobem és que el programa intenta registrar primer cada caràcter que troba, incloses les lletres que formen cada paraula. Haurem de fer servir una instrucció del mòdul string que aïlli col·leccions de caràcters separats per espais, paraules, per crear una llista: `words = re.findall(r'\w+', tweet).` El segon problema és que la llista discrimina majúscules i minúscules. Amb el codi `term.lower() for term in words` solucionem aquest problema: estem demanant a la màquina que apliqui la instrucció `term.lower`, que converteix tots els caràcters alfanumèrics en minúscules, sobre la llista words, que conté paraules i descarta lletres soltes.

El tercer problema són les paraules buides: aplicant un bucle if com el de l'anterior programa, finalment obtindríem una llista formada per paraules, però moltes d'elles són articles, adverbis i preposicions que no tenen sentit sense el seu context. En processament informàtic de llenguatge natural, aquestes paraules comunes que no tenen contingut s'anomenen paraules buides[22] i es filtren abans de començar els càlculs.

El quart problema són les paraules que no ens interessen del nostre codi. Ens trobem amb un conjunt curt de paraules que no ens interessen, algunes són pròpies de Twitter i la comunicació via internet en general ('rt', 't', 'https', 'co'), altres són els noms dels líders dels partits ('albert_rivera', 'pablo_iglesias_'), n'hi ha que són reiteratives amb la llista de

[22] Rajaraman, A.; Ullman, J. D. (2011). "Data Mining" a Mining of Massive Datasets, pàg. 1–17. Consultat el 2016 a http://i.stanford.edu/~ullman/mmds/ch1.pdf

paraules que ja sabem que contenen les piulades ('catalunya', 'españa', 'españoles'), i altres que senzillament han escapat a la llista de paraules buides ('va').

Per fer front a aquests problemes, necessitem una manera de filtrar tant paraules buides com paraules que no ens interessen. Per fer-ho, hem descarregat el mòdul stop_words que permet creuar una llista de Python amb un arxiu .txt que conté paraules buides amb una sentència `not in`. Hem descarregat un arxiu de paraules buides en castellà (ja que tots els nostres usuaris piulaven en castellà). Seguidament, hem creat una llista personalitzada de paraules que ens interessava filtrar, que hem anomenat `stop_tris`. Finalment, hem creat una funció (`get_tag_list`) que aplica tots aquests filtres a una cel·la.

```
from stop_words import get_stop_words
from stop_words import safe_get_stop_words
from stop_words import AVAILABLE_LANGUAGES
import string

stop_tris = ['cataluña', 'rt', 't', 'https', 'co', 'españa', 'albert_rivera','marianorajoy','catalán','pablo_iglesias_'
STOPWORDS = get_stop_words('spanish') + stop_tris

def get_tag_list(tweet):
    import re
    words = re.findall(r'\w+', tweet)
    return [term.lower() for term in words if term.lower() not in STOPWORDS and len(term) > 3]
```

Crear una llista tags amb el conjunt de paraules filtrades a totes les piulades

El codi que de la imatge té com a objectiu crear una llista d'elements del CSV com havíem fet amb la llista de partits polítics. Seguim el mateix principi que llavors: declarem una llista buida, que s'anomena `tags`. Dins del CSV, creem un bucle que ignori cel·les buides i *headers* i l'apliquem a la `row[2]`, creant una llista anomenada `tweet`. Li demanem que apliqui la funció `get_tag_list` (la funció que descarta lletres soltes, converteix en minúscula i filtra pels nostres 2 diccionaris), i el resultat queda inclós a una llista anomenada `lista_tags`. Finalment, demanem que vagi afegint el resultat a la llista tags a través de la instrucció `extend`. Amb la instrucció `print`, demanem a la màquina que mostri el nombre d'elements únics que conté la llista tags. Quan executem, el resultat és `99.726` paraules úniques no buides.

Ja només ens queda veure la freqüència amb la qual es repeteixen, és a dir, quantes vegades els partits polítics han fet servir cada una de les paraules. Només ens serà d'utilitat si recuperem les més usades, i per això farem servir un codi que ordena els resultats i mostra els 100 amb les freqüències més altes.

```
def top_tags(tag_list, top=100):
    freq = Counter(tag_list)
    print ("Top", top, "tags:")
    for tag, nb in freq.most_common(top):
        print(tag, "ha sigut utilitzat", nb, "cops")
    return freq.most_common(top)
top_tags(tags, 25)
```

Funció que retorna els termes més utilitzats de la llista `tag_lists`

El codi és molt simple: a la primera línia, declarem una funció que s'anomena `top_tags`, que opera amb els parametres `tag_list` (la llista sobre la que treballa) i `top=100`, un paràmetre per la instrucció posterior `most_common`. A la segona línia, creem un element anomenat `freq` que conté el resultat d'aplicar el comptador `Counter` a la llista `tag_list`. El resultat serà un diccionari que conté cada tag emparellat amb el nombre de cops que apareix (aquesta xifra rep el nom genèric d'`nb`). `Tag` es converteix, per tant, en un diccionari. A la següent línia, creem un bucle for pels elements `tag` i `nb` (paraula i nombre de cops que apareix) del diccionari `freq`, aplicant la instrucció `most_common(top)` (aquesta instrucció busca les X quantitats més altes, on X correspon al paràmetre `top` de la funció). Amb la funció `print`, introduint la frase "ha sigut utilitzat" i "cops" entre els parametres, la consola mostrarà els resultats:

```
adacolau ha sigut utilitzat 1484 cops
xavipabloada ha sigut utilitzat 1349 cops
país ha sigut utilitzat 1265 cops
madrid ha sigut utilitzat 1148 cops
inesarrimadas ha sigut utilitzat 994 cops
l6ncampaña20d ha sigut utilitzat 902 cops
problema ha sigut utilitzat 897 cops
gente ha sigut utilitzat 883 cops
creemos ha sigut utilitzat 704 cops
partidos ha sigut utilitzat 703 cops
queremos ha sigut utilitzat 680 cops
independentistas ha sigut utilitzat 662 cops
gracias ha sigut utilitzat 659 cops
fraternidad ha sigut utilitzat 653 cops
directo ha sigut utilitzat 643 cops
iglesias8aldia ha sigut utilitzat 638 cops
yovotopp ha sigut utilitzat 622 cops
caraacaral6 ha sigut utilitzat 622 cops
unidad ha sigut utilitzat 604 cops
xavierdomenechs ha sigut utilitzat 586 cops
hecho ha sigut utilitzat 576 cops
partido ha sigut utilitzat 522 cops
fuerza ha sigut utilitzat 511 cops
resolver ha sigut utilitzat 506 cops
neweconomyforum ha sigut utilitzat 487 cops
```

Codi 3 Repartició de paraules per usuaris

Fent servir els mateixos esquemes, hem fet un programa que ens indica quin són els 25 tags (paraules) més utilitzades per cada partit, i hem esbrinat que no tots els partits fan servir les mateixes. El que ens interessa ara es veure, sobre el total de paraules utilitzades, quan les ha fet servir cada partit. D'aquesta manera, esbrinem quins han estat els conceptes més emprats a Twitter, i de quina manera hi ha contribuït cada partit.

```
repart = {}
for tag, freq in top_tags(tags, top=10):
    repart[tag] = {}
    for partido, tags_list2partido in tagsXpartido.items():
        if tag in tags_list2partido:
            repart[tag][partido] = tags_list2partido.count(tag)
        else:
            repart[tag][partido] = 0
```

Codi de la repartició de tuits més freqüents per usuari

El primer que fem és crear una llista anomenada `tuitsXpartido` que contingui tots els tuits de cada partit. Per fer-la, hem seguit un procediment essencialment igual que per fer els diccionaris de partits i de paraules filtrades. Després, hem creat un diccionari anomenat `repart` que prenetem omplir amb la repartició de tags per usuari. Per fer-ho, hem creat un bucle for que passa per cada una de les 8 llistes dels partits i amb el condicionant `if` esbrina si les 10 més usades en general hi són. Després compta quantes vegades apareixen, i les afegeix al diccionari `repart`.

3.4 Visualitzacions

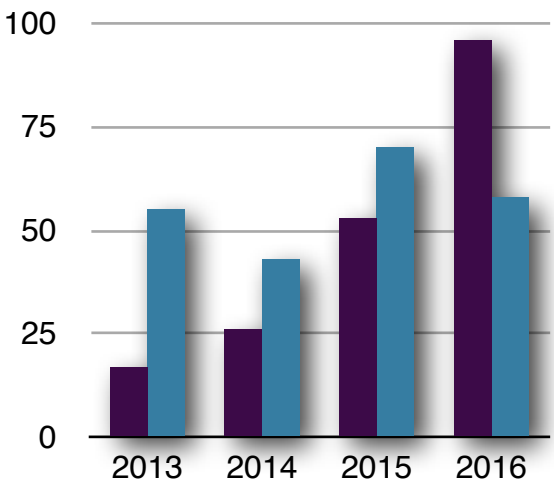
Alguns dels resultats del codi són fàcils d'entendre amb un simple cop d'ull, com ara la quantitat de piulades que ha fet cada partit. Però hi ha altres casos, com llistes de les 100 paraules més usades, que requereixen una explicació visual. Per cada una de les 4 seccions, hem dut a terme una visualització que ens ajuda a extreure conclusions.

Eines emprades: Existeixen diferents maneres per traduir resultats d'operacions en gràfics a través de Python. La que té més recorregut i eines és MatPlotLib, és potent i complexa i adreçada a usuaris avançats[23]. Pel propòsit del nostre exercici, ens hem decantat per PlotLy. Aquesta eina té quatre avantatges clau: és gratuïta, és simple de fer servir, els gràfics per defecte són potents i interactius i és fàcil d'integrar en HTML, amb la qual cosa juga amb avantatge a l'hora d'integrar-se a la pàgina d'un mitjà de comunicació.

3.4.1 Tipus de gràfics

Existeixen diferents tipus de gràfics que compleixen diverses funcions. S'elegeix un tipus o un altre en funció del tipus de dades que s'hi representen[24]. Aquestes poden ser qualitatives o quantitatives. Les primeres són aquelles que no es poden expressar numèricament, dividides en ordinals si segueixen un ordre o categòriques si no ho fan. Les segones, les quantitatives, són les dades numèriques, que poden ser discretes si prenen valors enters o contínues si prenen qualsevol valor a dins d'un interval.

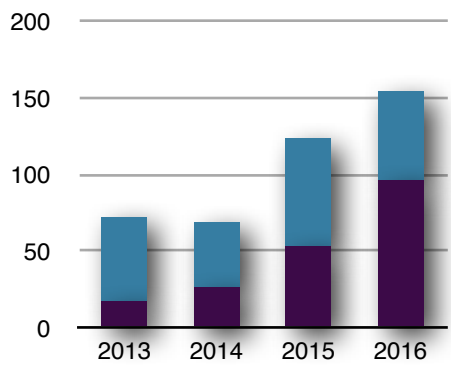
1) Barres



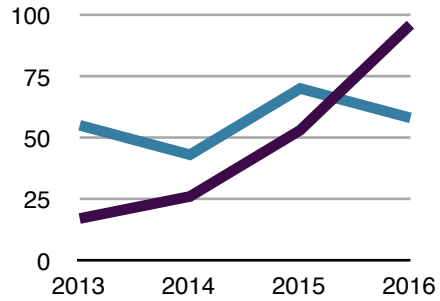
Es fan servir per expressar una variable quantitativa discreta. Serveixen per comparar magnituds de diferents categories, o per veure l'evolució en el temps d'una magnitud. Si dividim una variable continua en intervals, també podem representar-la en un gràfic de barres. Poden ser senzills, agrupats (el que acompanya aquestes línies), o apilats. Podem posar l'eix al mig per obtenir un gràfic bidireccional, com les piràmides de població.

[23] Moffitt, Chris (20/01/2015). Overvirew of Python Analysis Tools. *Practical business python,* consultat el 2016 a http://pbpython.com/visualization-tools-1.html

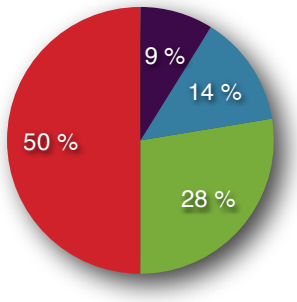
[24] Font: Instituto nacional de estadística. "Tipos de gráficos, ¿cuál uso?", consultat el 2016 a http://www.ine.es/explica/docs/pasos_tipos_graficos.pdf



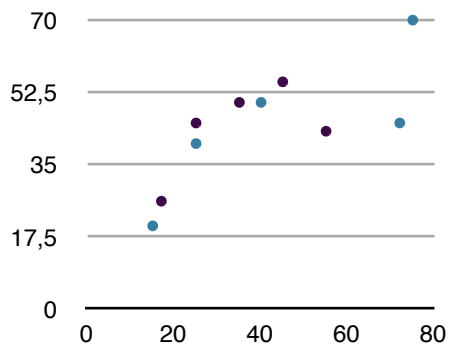
Existeixen de tres tipus: simples, agrupats, o apilats. Els simples representen una única sèrie de dades. Els agrupats tenen grups de dades que comparteixen color. El **gràfic apilat**, com el que acompanya aquestes línies, conté varies sèries de dades. La barra es divideix en segments de diferents colors, i cada un d'ells representa una sèrie.



Línies: es tracta d'una representació gràfica de la relació que existeix entre dues variables, amb la intenció de reflectir els canvis produïts. És especialment útil per reflectir l'evolució d'una variable durant un determinat període de temps.



De sectors: és una representació circular de les freqüències relatives d'una variable qualitativa o discreta, que permet fer una comparació ràpida i senzilla. És útil quan hi ha poques variables.



De dispersió: Mostra en un eix cartesià la relació entre dues variables. La seva funció és mostrar el grau de correlació entre dues variables: mostra si l'increment dels valors d'una variable (la independent, generalment a l'eix horitzontal) altera d'alguna manera els valors d'una segona variable, dependent, al vertical. Segons la forma i direcció del núvol de punts es pot deduir el tipus de relació.

3.4.2 Elecció del tipus de gràfic

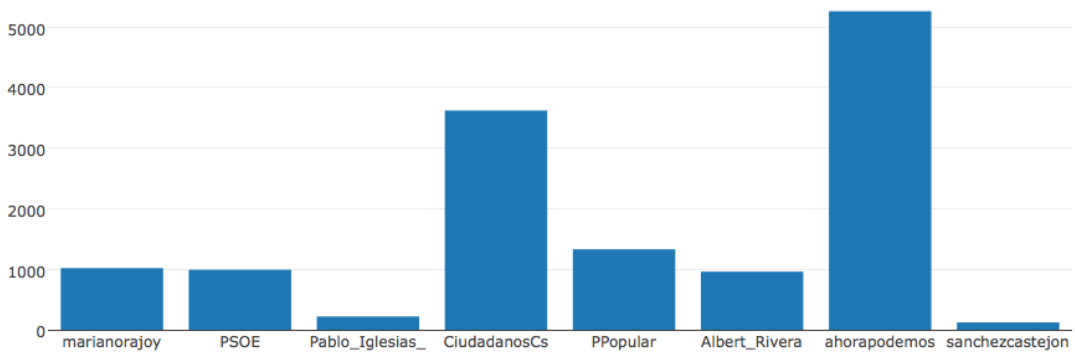
Les nostres dades són quantitatives discretes, no segueixen cap evolució temporal i són un nombre relativament alt de variables (8 usuaris). De les 4 seccions del nostre codi, les tres primeres (nombre de piulades segons usuari, freqüències de les 10 paraules clau, paraules clau de cada partit), la decisió clara és optar per un gràfic de barres. El quart apartat, la repartició del top de paraules clau segons el partit que les ha emés, requerirà un gràfic més complexe. En aquest cas, la variable es divideix en fins a 10 sèries de dades – paraules clau – i per tant, l'opció correcta és el gràfic de barres apilades.

3.4.3 Codi de les visualitzacions

La integració de plotly a python és extremadament simple. Com tenim les dades que ens interessen emmagatzemades correctament en diccionaris i llistes, només cal que importem plotly, declarem la llista `data` que conté la instrucció `go.bar` per fer un gràfic de barres. Dins del seu codi, indiquem que les claus i els valors del diccionari d'usuaris o piulades han de ser assignades als eixos x i y, per horitzontal i vertical. Plotly retorna un gràfic de barres fàcilment integrable en HTML i interactiu, que mostra la quantitat exacta de piulades quan fem passar el cursor per la barra.

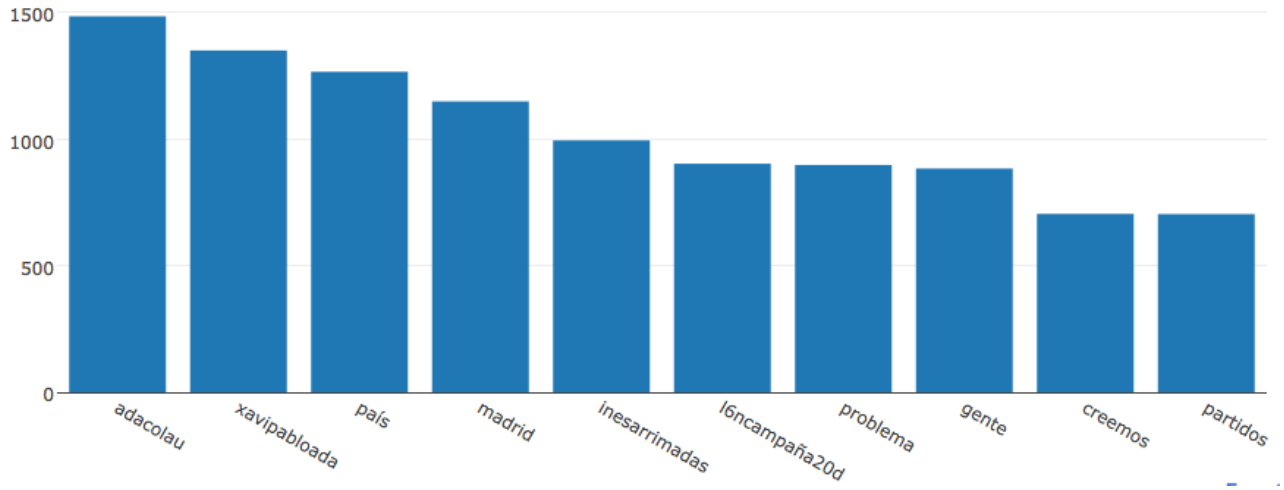
```
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.plotly as py

data = [
    go.Bar(
        x=[str(n) for n in partidos.keys()],
        y=[str(n) for n in partidos.values()],
    )
]
iplot(data)
```



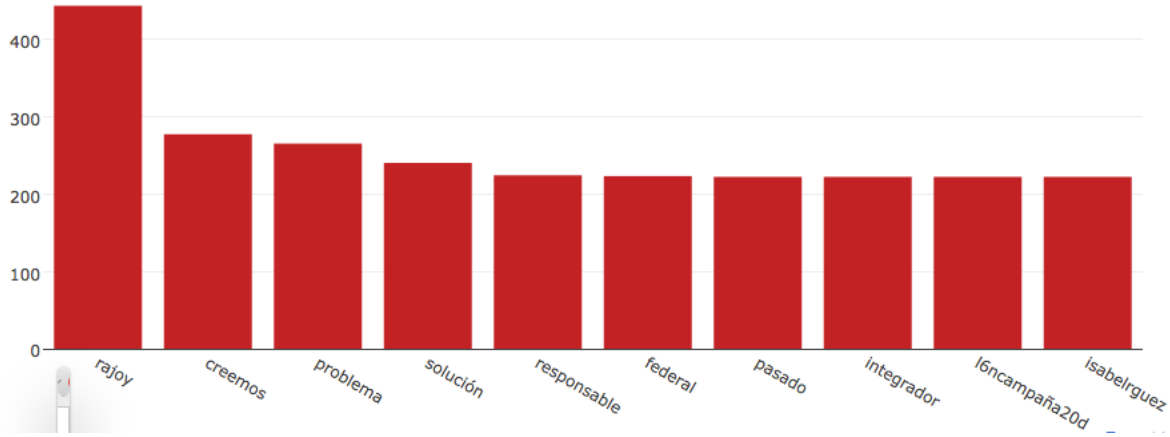
Codi i gràfic de distribució de piulades segons usuari

1 Paraules clau: les 10 paraules sobre Catalunya més usades pels partits polítics

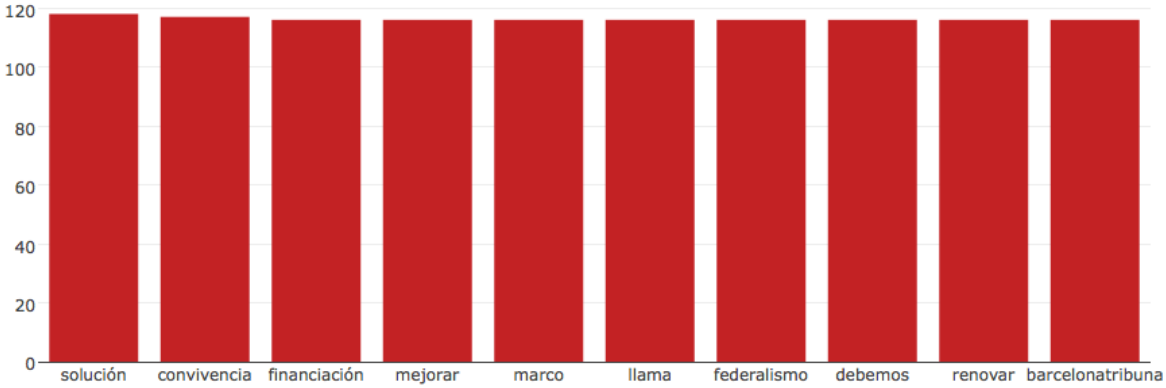


2 Paraules clau de cada partit polític i líder del seu partit

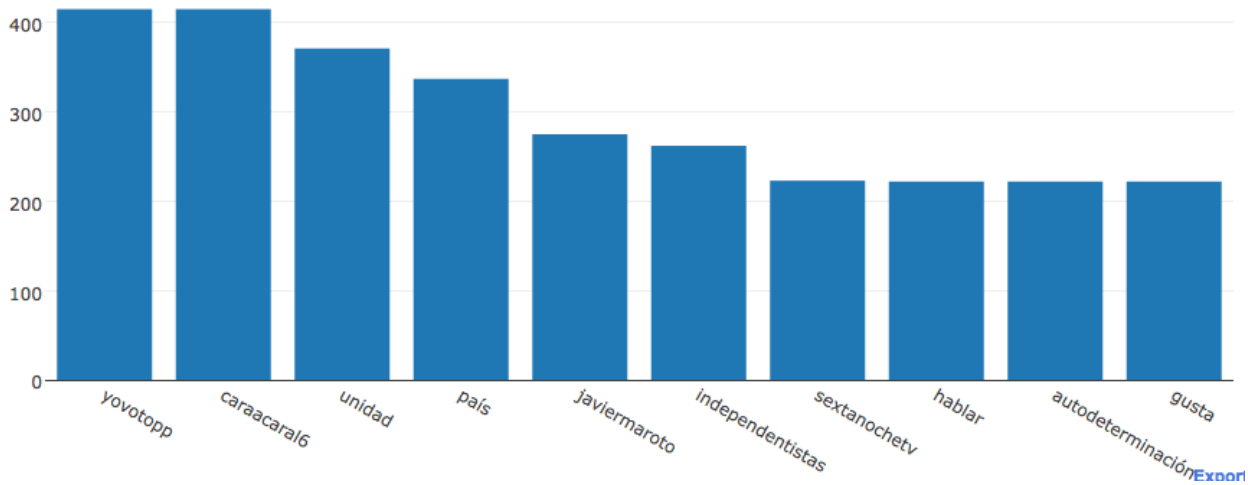
1. **PSOE**, sobre un total de 992 piulades



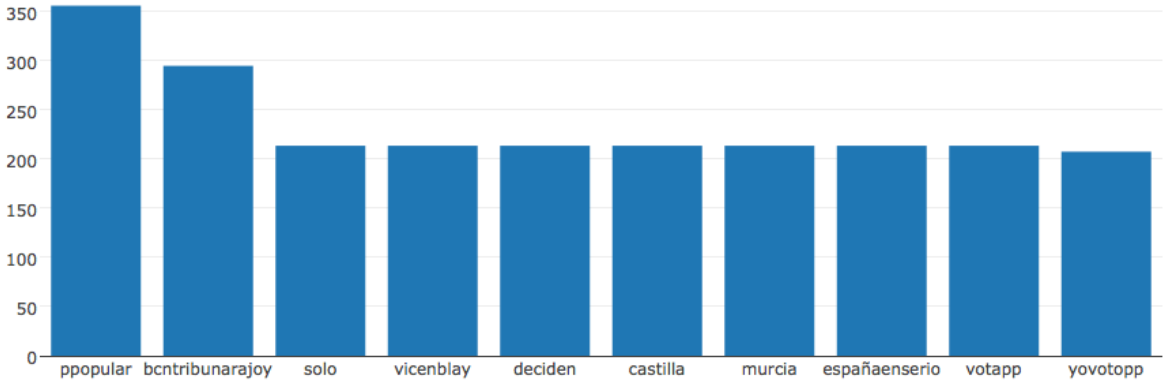
2. **Pedro Sánchez**, sobre un total de 122 tuits



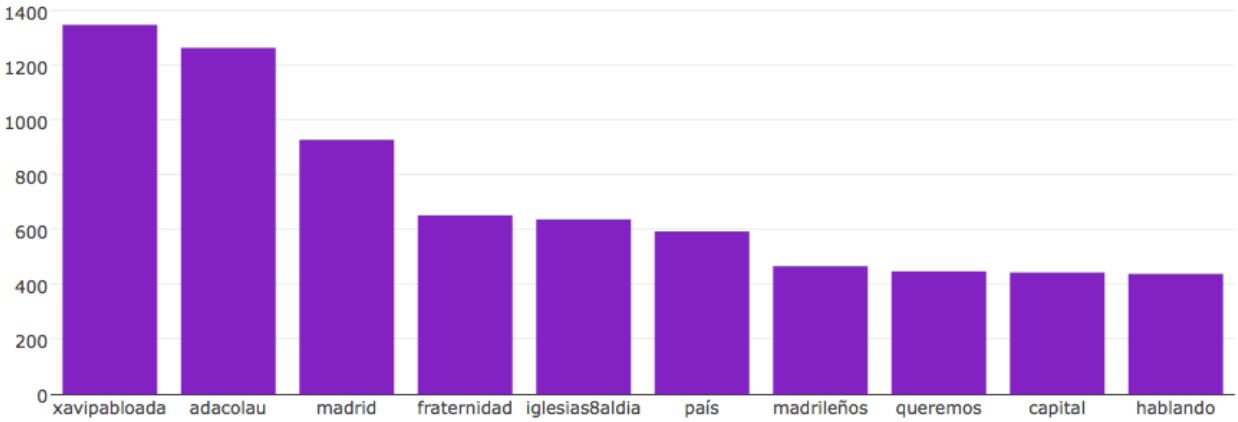
3. **Partit Popular**, sobre un total de 1329 tuits



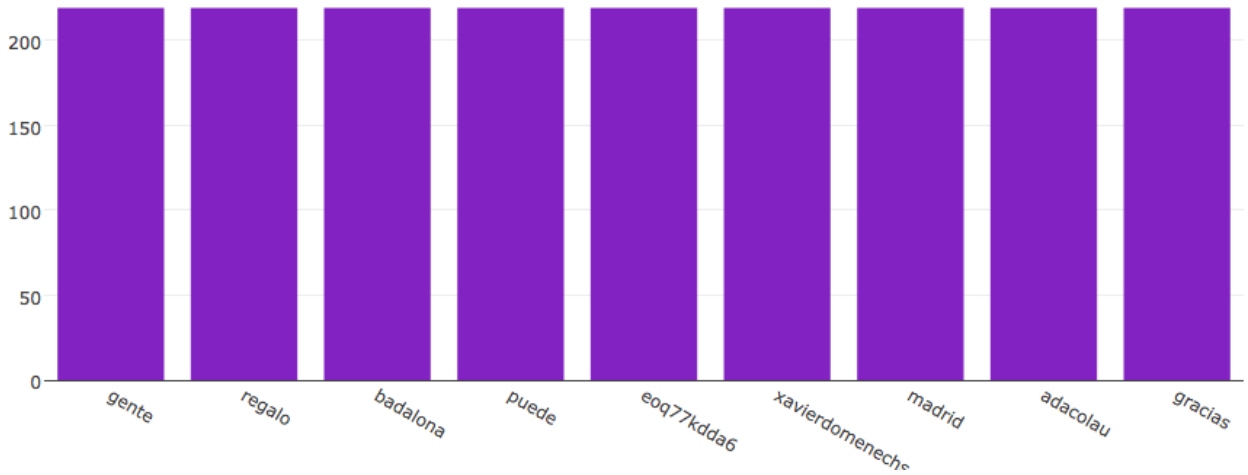
4. **Mariano Rajoy**, sobre un total de 1018 piulades



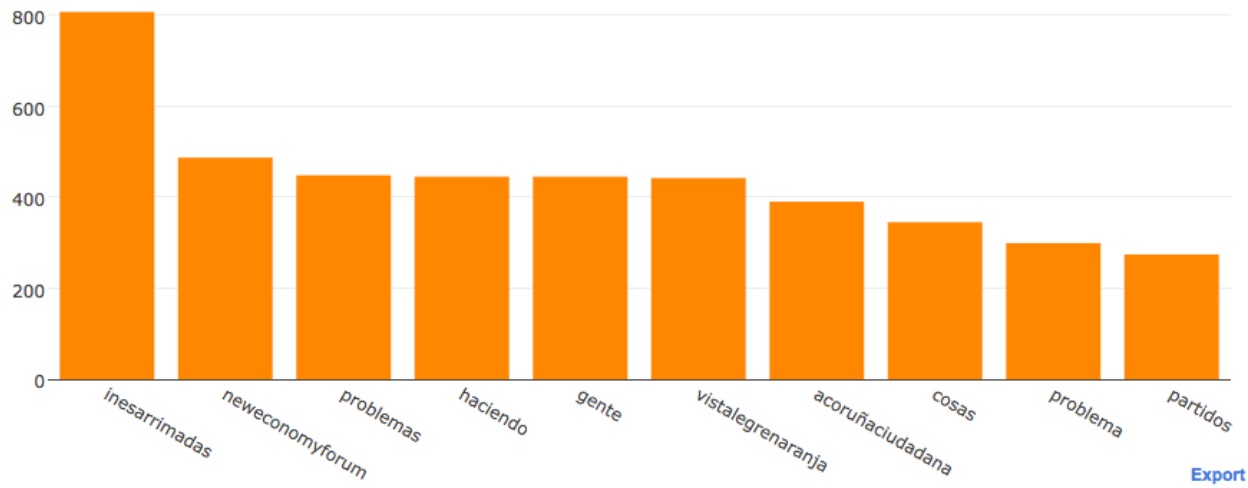
5. **Podemos**, sobre un total de 5259 missatges



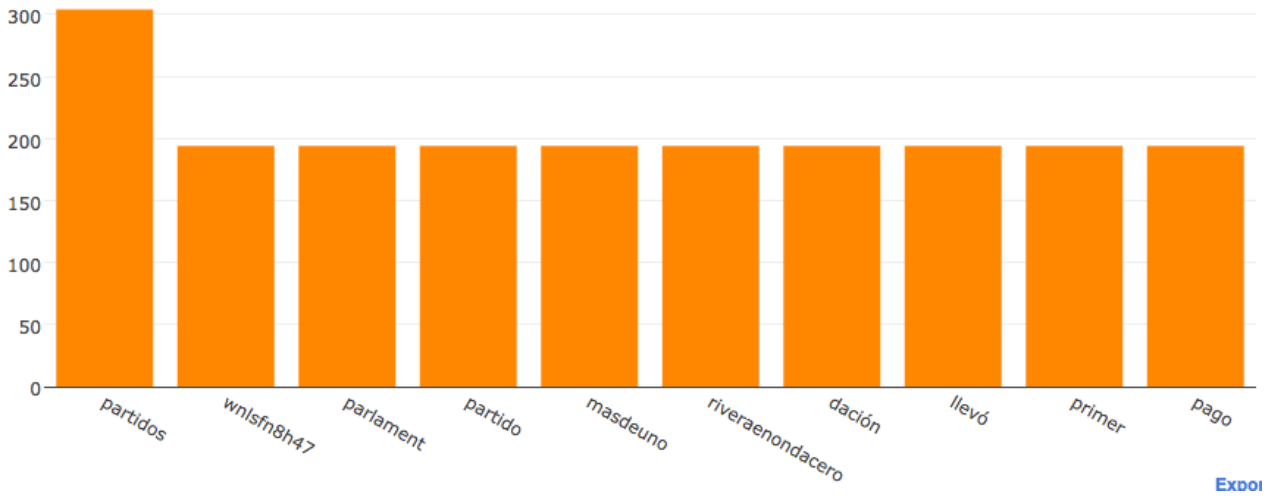
6. **Pablo Iglesias**, sobre un total de 219 tuits



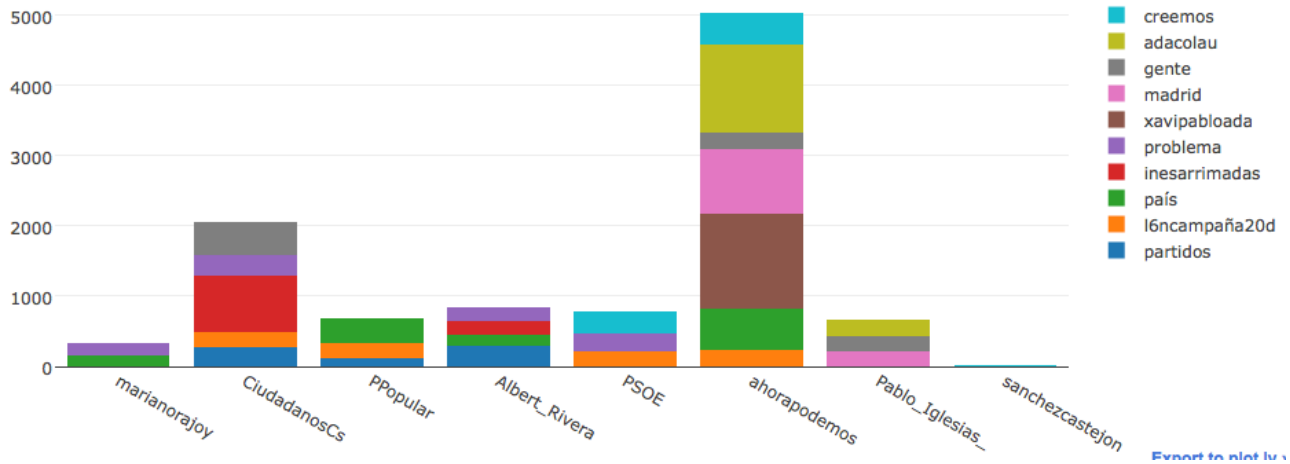
7. **Ciutadans**, sobre un total de 3620 missatges



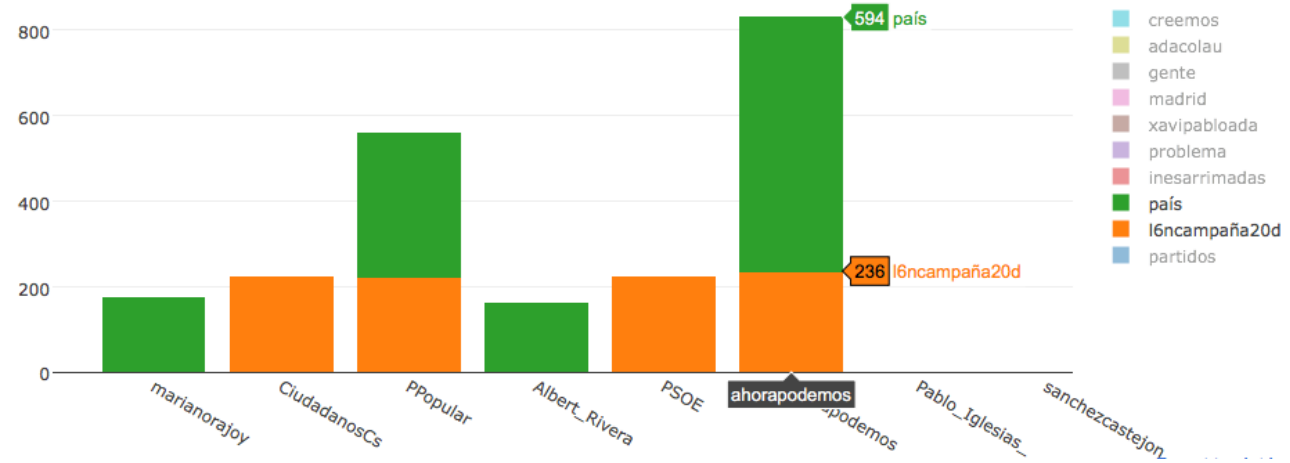
8. **Albert Rivera**, sobre un total de 960 piulades



3 Gràfic final: Repartiment de paraules clau per usuari



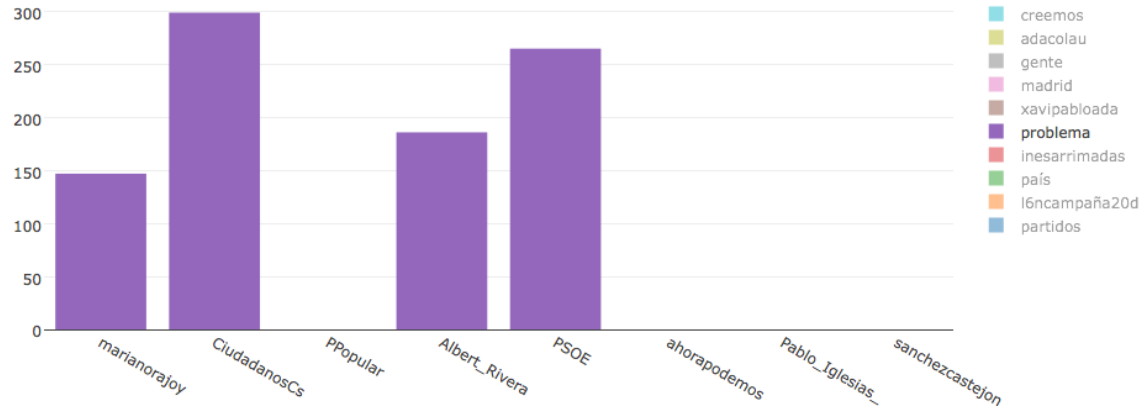
Aplicant plotly al codi que mostrava el repartiment de paraules clau per usuari, obtenim la visualització final, de tipus gràfic de barres apilades. D'aquesta manera, podem copsar a cop d'ull quins són els usuaris que han fet més piulades, i quines han estat les paraules clau. El gràfic és interactiu, mostra el total de cops que s'ha fet servir cada paraula clau i fer zoom i moure's pel gràfic. A més, té un element molt important: la possibilitat d'activar i desactivar les paraules clau que es visualitzen. Això permet extreure tota una sèrie de conclusions d'interès periodístic.



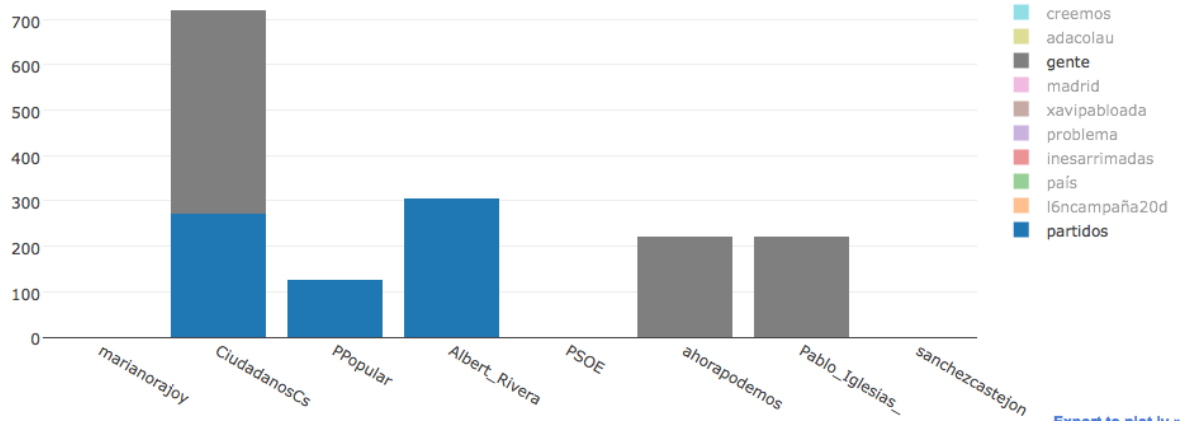
Aquest és un exemple de dos tags activats ('país' i 'l6ncampaña20d') i l'efecte que es produeix quan deixem el cursor a sobre d'una barra, i el programa mostra el nombre de piulades.

3.5 Anàlisi periodístic

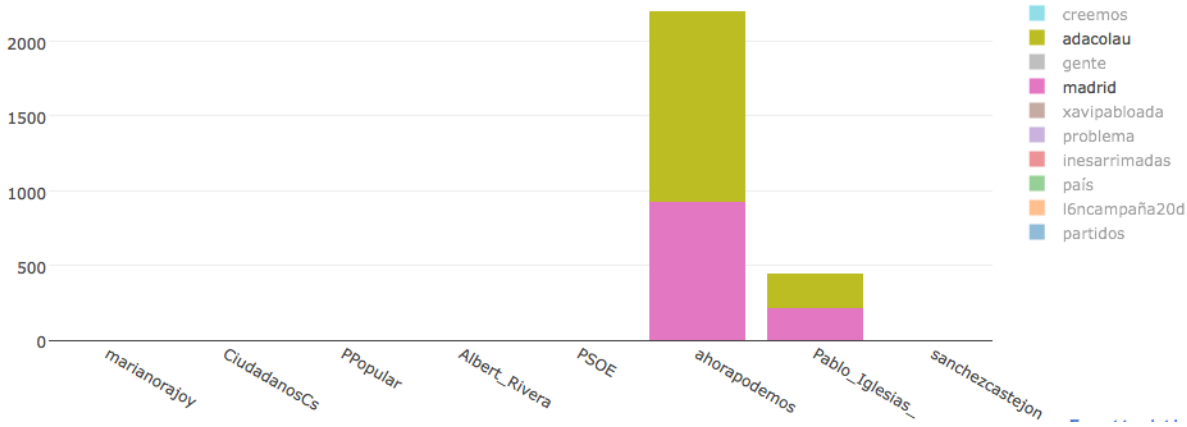
D'entre totes les conclusions que es poden extreure d'aquestes dades, la més destacada és que Rajoy, PSOE, Ciutadans i Albert Rivera parlen de Catalunya en conjunció amb la paraula "problema" més cops que cap altre partit. Les dades es mostren a continuació:



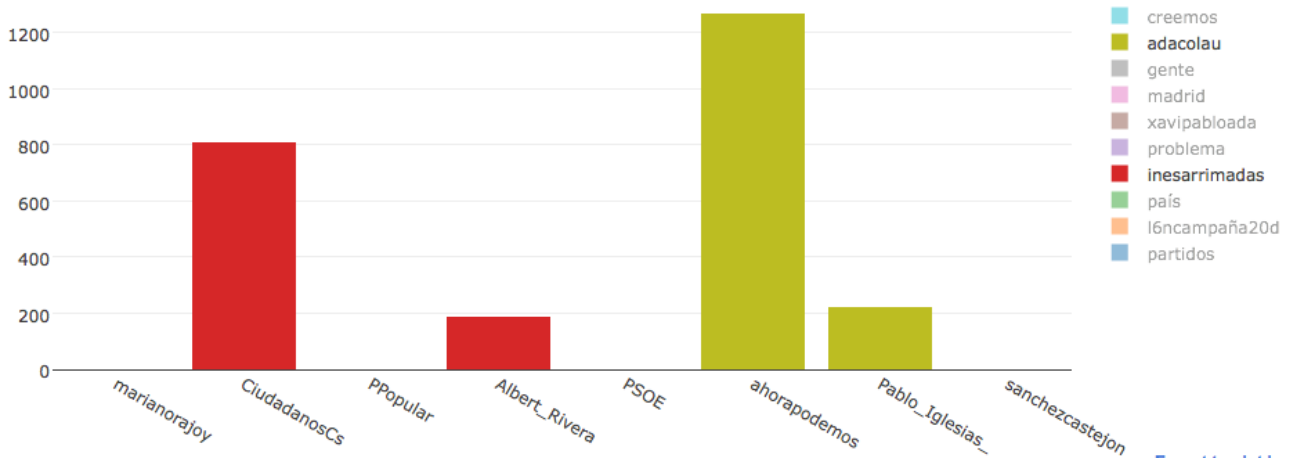
En segon lloc, els nous partits són els que fan servir la paraula 'gente' quan fan referència a qüestions sobre la independència de Catalunya. PP i Albert Rivera, en canvi, parlen de 'partits', i el compte de Ciutadans fa servir les dues expressions.



Podemos ha fet mencionat Madrid prop de 1000 cops, i Ada Colau més de 1200:



En últim lloc, Ciutadans i Podemos han mencionat les seves cares visibles a Catalunya prop de 800 i 1200 cops, respectivament.



Per tant, extraiem les següents conclusions:

1) Per PSOE, PP i Ciutadans, existeix un problema amb Catalunya i l'autodeterminació
2) Els partits del canvi apel·len a "la gent" quan parlen del tema
3) Podemos ha fet un ús extensiu de les seves figures mediàtiques, conegudes com a 'alcaldesses del canvi', a l'hora de parlar de Catalunya
4) Podemos i Ciutadans esmenten amb molta freqüència les cares visibles de les seves organitzacions al territori català, Ada Colau i Inés Arrimadas.

De les visualitzacions anteriors a aquestes, hem extret les següents conclusions:

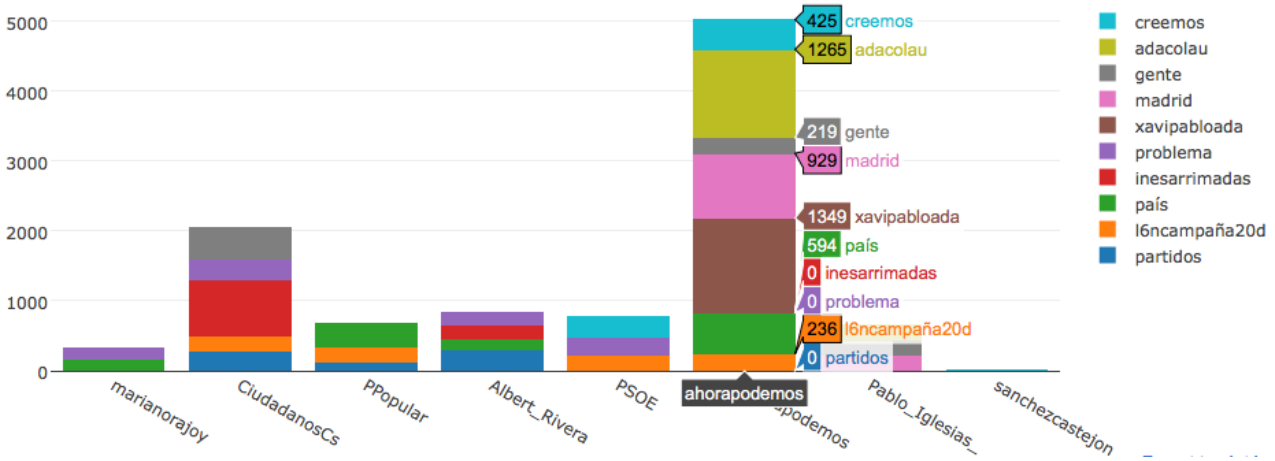
1) Els qui més han parlat del tema han estat Ciutadans (3600 missatges) i Podemos (5200). Els qui menys, Pablo iglesias i Pedro Sánchez, amb 220 i 120 respectivament. La resta de partits se situen al voltant dels 1000 missatges sobre Catalunya.
2) Les dues paraules clau que hem trobat per cada partit són: pel PP 'unidad' i 'pais', pel PSOE 'problema', 'solución' i 'federalismo'; 'fraternidad' i 'pais' en el cas de Podemos, i 'problema' i 'gente' per Ciutadans.
3) En referencia a Catalunya, tan Rajoy com Sánchez han piulat quasi exclusivament amb el hashtag d'un esdeveniment concret (BCNtribuna, una conferència organitzada per la Vanguardia el 17 de desembre en el cas de Rajoy y el 18 en el de Sánchez). Iglesias va piular només sobre el míting de Badalona del 12. Rivera va piular molt durant la seva aparició a Onda Cero el 15 de desembre però ha estat més prolífic a la xarxa social.

A partir d'aquestes dades, i aportant les de context, es pot fer un breu article que acompanyi les dades i expliqui els elements clau que s'hi poden trobar:

Catalunya, un problema per tots menys per Podemos a Twitter

En relació a Catalunya i l'autodeterminació, els partits han parlat de "problema" però també de "solució" a les seves piulades – Els nous partits apel·len a la "gent" i a les seves figures clau

TRISTAN IBÁÑEZ - 18/12/2015 Durant la campanya electoral que acaba avui, els partits contraris a l'autodeterminació han deixat clar a la xarxa social de l'ocell que les coses estan lluny d'estar arreglades a Catalunya. Així ho demostren les dades recollides i analitzades per aquest mitjà, que reflecteix que han mencionat la paraula 'problema' en un de cada cinc tuits sobre Catalunya, autodeterminació o independència. Podemos, que no l'ha fet servir, ha sigut el partit que més cops ha piulat sobre Catalunya.



Gràfic interactiu de les paraules clau utilitzades a Twitter pels partits i els seus líders / TRISTAN IBÁÑEZ

CARES CONEGUDES El partit lila ha parlat freqüentment de 'gent', paraula que també ha sobresortit per la seva freqüència als missatges de Ciutadans. Però els seus grans punts de referència al principat són Ada Colau i Inés Arrimadas, els pesos pesants dels partits del canvi a Catalunya, que han sigut mencionades en prop d'un terç dels tuits dels seus partits. Podemos està intentant treure rèdit de les victòries de les alcaldesses del canvi, que apareixen sovint a les seves participacions a la xarxa social.

El partit de Rajoy, que ja ha qualificat d'"experiments populistes" als emergents en una entrevista en període de campanya, ha sigut molt més reservat que els seus nous rivals pel que fa al tema de l'autodeterminació catalana, sobre el que ha emès 1000 missatges. Podemos i Ciutadans no han sigut tímids en aquest sentit, i amb els seus 3500 i 5000 missatges demostren que els polítics del canvi se senten igual de còmodes a les xarxes socials que a un plató de televisió. L'estratègia donarà els seus fruits demà, durant les votacions que marcaran el punt final d'una campanya en què les xarxes socials hauran jugat un paper clau.

4 Anàlisi de viabilitat

En aquesta secció analitzarem quin seria el cost d'incorporar periodisme de dades per un mitjà de comunicació que tingui presencia a la web. En primer lloc, llistarem tots els recursos humans, materials i de temps que són necessaris i els seus costos, després esbrinarem com és el cicle productiu d'un exercici com aquest, i ferem un cop d'ull a la inversió inicial que cal. Amb aquestes dades, proposem una planificació horària en la que detallem els processos del cicle productiu i com es reparteixen durant una jornada laboral.

4.1 Recursos:

Humans: com hem vist a l'apartat 2.2.1 "Un equip de periodisme de dades tipus", l'equip ideal està format per tres professionals. Per començar, un programador, que pugui dur a terme anàlisi de dades i tasques tècniques darrere de la visualització de dades i la integració a la plataforma en línia del mitjà de comunicació. En segon lloc, un periodista que tingui coneixement de l'actualitat i les rutines professionals per detectar quines són les àrees de l'actualitat on buscar informació rellevant, i en tercer lloc un dissenyador, que tingui nocions de comunicació visual i jerarquia de la informació per poder comunicar el que l'anàlisi de dades ha revelat de manera òptima. El periodista de dades reuneix els coneixements necessaris en aquestes tres disciplines per dur a terme una peça com la que hem produït sense l'ajuda de cap altre professional. El cas de la periodista de dades catalana Eli Vivas, dissenyadora gràfica de formació, és un exemple de professional interdisciplinària.

Eines

Físiques: La unica eina física necessària és un ordinador connectat i un monitor. És cert que el tractament de *big data* amb fulls de càlcul potents com Excel pot ser molt exigent sobre la màquina, però hem recomanat LibreOffice, molt més indulgent amb els sistemes, i si tractem les dades des de python, la càrrega sobre el hardware és mínima. Per posar un exemple, aquest treball s'ha confeccionat amb un ordinador iMac 27" de 2011 amb un processador de 8 nuclis però només 4Gb de RAM, i tots els bucles que hem fet servir i que analitzen un per un els 13.000 tuits del CSV retornen resultats en fraccions de segon. Les visualitzacions de dades són una mica més pesades però tot i això el temps de processament és obviable: amb aquesta màquina, els gràfics tarden aproximadament 10 segons a generar-se. El tractament del CSV amb Numbers o Excel penalitza sensiblement el sistema, però amb LibreOffice el funcionament és idoni.

En cas que el mitjà no disposés d'ordinadors, proposem el model Acer Aspire XC-703_WJ2900, que té prestacions equivalents a la màquina que hem fet servir per dur a terme la peça: un processador de 4 nuclis de 2.66Ghz, 4Gb de ram i un disc dur d'1Tb. El seu preu sense IVA és de 245.09 € a la botiga PCGreen de Barcelona. Com monitor, seria adequat un model com el B206WQL de la mateixa marca, de 19,5", resolució de 1440 x 900 i amb un preu sense IVA de 101,32 € a la mateixa botiga.

Software: Pel que fa al software, hem fet servir les següents eines, totes elles gratuïtes:
Compte de developer a Twitter - gratuït (https://dev.twitter.com)
API de twitter: TAGS - gratuït (http://tags.hawksey.info)
Google Spreadsheets - gratuït (http://docs.google.com)
Full de càlcul per treballar amb CSV: libreoffice - gratuït (http://www.libreoffice.org)
Neteja i tractament de *big data:* Google Open Refine - gratuït (http://openrefine.org)
Llenguatge de programació Python: gratuït (https://www.python.org)
Entorn gràfic Jupyter: gratuït (http://jupyter.org)
Móduls de Python:

CSV: gratuït, integrat al sistema
Counter: gratuït, integrat al sistema
re: gratuït, integrat al sistema
String: gratuït, integrat al sistema
Stop_words: gratuït, descarregar des de https://pypi.python.org/pypi/stop-words
plot.ly: gratuït, amb plans de pagament per compartir els gràfics via el seu núvol, que en el nostre cas no són necessaris (http://www.plot.ly)

Altres: navegador Safari i Chrome (gratuït)
Sistema operatiu: MacOS X (gratuït) – totes aquestes eines són multiplataforma i es poden fer servir tant en Windows 10 (gratuït) com a les diferents distribucions de Linux (gratuïtes).

Temps:
Recollida de dades: el temps total del nostre exercici cobreix les dues setmanes de la campanya, però depèn de la durada de l'esdeveniment que decidim cobrir. L'ideal seria disposar de diferents fulls de TAGs en els que fem el seguiment simultani de diferents esdeveniments públics, de tal manera que sempre disposem de diferents bases de dades amb les què treballar. En aquest sentit, cal tenir en compte les restriccions imposades per Twitter per protegir els seus servidors, i que permeten a cada API autenticada dur a terme

fins a 180 cerces per quart d'hora[25]. Superar aquest límit implicaria una penalització de 15 minuts durant els quals es nega l'accés a la API. Imaginem que estem seguint quatre temes. Si configurem un *scrapping* a TAGs perquè busqui dades cada hora, i el programem perquè cada una d'elles ho faci amb un quart d'hora de diferència de la següent, tindríem la màxima optimització segons les limitacions de Twitter. Així, la primera cerca començaria, per exemple, a les 10h00, la segona a les 10h15, la tercera a les 10h30 i la quarta a les 10h45, i després s'anirien repetint un cop cada hora.

En tot cas, un cop estem familiaritzats amb les eines, instal·lar i programar les API per recollir les dades és un procés que es pot fer en uns 15 minuts, i comprovar el seu funcionament i actualitzar les dades i fer còpies de seguretat es pot fer un cop al dia en no més de 15.

4.2 Inversió inicial i cicle de producció

El procés inicial de presa de decisió sobre quin tema i amb quines eines el tractarem el situem en al voltant d'1 hora. Pel que fa a la programació per analitzar les dades, en python, cal dir que el codi utilitzat per escriure aquestes línies es podria adaptar per reutilitzar-lo amb altres dades i això faria reduir el temps necessari. De totes maneres, un cop tenim les dades i una primera idea del que busquem en elles, per un programador professional o un periodista de dades amb nocions avançades de python, la tasca no hauria de durar més d'entre tres i quatre hores. Un cop tenim les dades, fer les visualitzacions en una eina com plotly és una feina senzilla per un usuari expert, i calculem que entre l'elecció del tipus de dades i la confecció del codi ens prendria entre una i dues hores. L'anàlisi posterior i la confecció de la peça és un procés d'entre dues i tres hores.

Inversió inicial:

La inversió inicial en recursos humans serien els cursos en disseny i programació que permetin a un periodista dur a terme les tasques d'anàlisi i visualització. Pel que fa al material, les hem xifrat en 346,41 € per l'ordinador que ens permetrà dur a terme les peces. En quant a software, el cost de la inversió és nul perquè totes les eines són gratuïtes. En temps, instal·lar tot el programari és una inversió major del que pot semblar. Els codis de programació estan instal·lats de manera molt profunda als sistemes operatius, i és habitual que hi hagi problemes amb la localització dels mòduls o la configuració de l'entorn gràfic, especialment si hi ha dues versions del mateix llenguatge.

[25] API Rate Limits. *Twitter / Developers*, consultat el 2016 a https://dev.twitter.com/rest/public/rate-limiting

Per posar un exemple, MacOS X té el llenguatge Python instal·lat per defecte a la seva versió 2.7. Si actualitzem a Python 3.5 sense assegurar-nos d'esborrar la versió anterior i tots els seus complements i mòduls, és molt possible que Jupyter no detecti la nova versió i segueixi funcionant amb l'anterior. Fins i tot és possible que algunes parts com els mòduls sí que quedin actualitzats, de tal manera que Jupyter intenta fer servir el codi antic amb els mòduls nous, que tenen una sintaxi lleugerament diferent, provocant errors i fent impossible treballar amb el llenguatge. Tots aquests arxius estan emmagatzemats a la llibreria del sistema, de difícil localització per l'usuari, i solen estar protegits, de tal manera que cal fer servir el terminal del sistema per localitzar-les i esborrar-les. Si no tenim una instal·lació neta de python, cada mòdul que instal·lem, com plotly, corre el risc d'ubicar-se automàticament en una localització errònia i generar errors. Amb tot això es dedueix que caldria no menys d'una jornada laboral completa per assegurar-se que tenim tots els programes, llenguatges, mòduls i complements correctament instal·lats al sistema.

El cicle de producció seria cíclic, ja que estem seguint 4 esdeveniments alhora, i per tant tenim accés a un flux constant de dades que ens permet completar tot el procés que se'n deriva de manera diària. De totes aquestes dades, es dedueix el següent cicle diari:

	Idea inicial	Configuració inicial TAGs	Anàlisi de dades	Descans	Visualització	Confecció de la peça final	Mantenim ent TAGs
Horari	09:00-10:00	10:00- 10:15h	10:15h- 13:15h	13:15h - 14:15h	14:15h - 16:15h	16:15h-17:45	17:45h-18h

Aquest és un cicle diari de 8h que permetria incorporar a la redacció un periodista de dades que produeixi una peça diària d'anàlisis de Twitter, amb la idea de treballar 5 dies a la setmana.

5 Conclusions

El periodisme de dades és una tècnica que ja incorporen molts grans mitjans internacionals, com ara el Guardian o el New York Times, i cada cop són més els que van creant seccions que els permetin extreure informació de les grans quantitats de dades que es poden obtenir d'Internet, com les que s'han derivat de casos com *Wikileaks* o dels "papers de Panamà". Això podria certificar l'èxit d'aquesta disciplina i pronosticar una ràpida acceptació als mitjans del nostre territori, però això sembla no estar-se complint. Si bé mitjans com BTV o el Confidencial estan obrint seccions de dades, altres històrics com el Pais s'estan mostrant molt menys disposats a fer-ho.

Durant la confecció d'aquest treball, hem pogut copsar quines són les dificultats que planteja l'exercici del periodisme de dades, claus per entendre de quina manera es pot agilitzar el seu procés d'incorporació als grans mitjans. Per un costat, les rutines de producció la converteixen en una disciplina que se sent més còmoda produint peces de llarg alè, amb una relació temporal més laxa amb l'actualitat, més pròximes al reportatge que a la notícia pura per expressar-ho en termes de gèneres convencionals.

El procés de recollida i anàlisi de les dades pot ser relativament ràpid si es repeteixen esquemes establerts, com el que es podria derivar d'aquest exercici, aplicable a posteriors anàlisis de lèxic a Twitter. Però aquest és un esquema molt precís, les possibilitats d'obtenir informació de qualitat són molt més amplies i cada una requereix un procés personalitzat segons les característiques de les dades a analitzar i el tipus d'informació que permetem fer amb elles.

A banda de la complexitat dels processos productius, cal tenir en compte dos elements que hem descobert amb aquest exercici. Per un costat, per definició, el *Big Data* presenta conjunts de dades de dimensions enormes, però també difícils de pronosticar. En el moment en el que vam començar a recullir dades, no sabíem quin seria l'abast del conjunt de missatges que reculliríem, ja que seguíem la seva emissió en directe. 13.000 piulades que només fan referència a Catalunya i només provenen dels partits polítics i només es van emetre durant la campanya ens sembla una xifra enorme, que no havíem esperat, i que a més demostra la quantitat d'informació que es genera diàriament a les xarxes socials.

Per l'altre costat, per molt que proposem una tesi inicial quan afrontem el projecte que intentem demostrar o rebatre amb la conclusió d'aquest, la veritat és que estem davant de fets impossibles de predir, i per tant les conclusions d'interès periodístic, o fins i tot la seva absència, no es poden començar a copsar fins ben entrat el procés de recollida i anàlisi. Cal tenir en compte, doncs, aquest factor d'incertesa, i la possibilitat que a vegades el procés d'anàlisi sigui estèril i haguem de descartar el treball dut a terme, amb la conseqüent pèrdua de recursos. En aquest sentit, és bàsica la intuïció periodística dels professionals, i el coneixement de la tècnica que només l'experiència poden proporcionar.

El periodisme de dades, però, és una tendència natural del periodisme, no només per la necessitat de trobat estratègies per digerir i explicar les notícies que deriven del *big data*, sinó per presentar una oportunitat de fer evolucionar i adaptar el discurs periodístic als nous usos de consum, derivats de les noves plataformes, impulsades per la generació del Mil·leni. Aquesta exigeix un nou tipus de comunicació, visual, tàctil i ràpida, i està disposada a sacrificar part de l'explicació que prové del text llarg a canvi de manipular contingut del que pugui extreure les seves pròpies conclusions. Tot i que el seu despertar és lent, el periodisme de dades pot ser un element clau a l'hora de construir aquest nou llenguatge comunicatiu, interdisciplinari i interactiu.

6 Annexos

Bibliografia:

Meyer, Philip (8/12/ 2011) Riot theory is relative. *The Guardian.* recuperat des de http://www.theguardian.com/commentisfree/2011/dec/09/riot-theory-relative-detroit-england

Meyer, Philip (2002) - *Precision Journalism: A Reporter's Introduction to Social Science Methods.* Pàg. 4 Oxford: Rowman & Littlefield

Shaw, Jonathan (2014). Why big data is a big deal. Harvard Magazine, des de http://harvardmagazine.com/2014/03/why-big-data-is-a-big-deal

Benjamin Howard, Alexander (2014) *The art and science of Data-driven journalism*. Nova York: Columbia School of Journalism

Ilustre Colegio de Abogados de Madrid, consultas jurídicas frecuentes, a http://crsa.icam.es/web3/cache/CRSA_asesora.html

En el mundo, ya hay más obesos que flacos (03/04/2016) *La Vanguardia,* a http://www.lavanguardia.com/vangdata/20160403/40847295800/obesidad-grafico-mapa-mundo-raking-paises.html

Gray, Jonathan; Chambers, Lucy; Bounegru, Liliana (2012) *The Data Journalism Handbook,* Sebastopol(EEUU): O'Reilly Media

Web API Code Examples & Libraries. *Spotify Developer,* consultat el 2016 a https://developer.spotify.com/web-api/code-examples/

Things every developer sould know, *Twitter / Developers*, consultat el 2016 a https://dev.twitter.com/overview/general/things-every-developer-should-know

Font: Twitter.com, pàgina per desenvolupadors, secció APIS, capítol "The search API" a https://dev.twitter.com/rest/public/search

Ventouris, Anastasios (2013) A Python guide for open data file formats. *Open Knowledge Foundation Labs,* consultat el 2016 des de http://okfnlabs.org/blog/2013/10/17/python-guide-for-file-formats.html

Hellerstein, Joseph M. (2008) *Quantitative Data Cleaning for Large Databases*, pàgina 1 Boston: UC Berkeley, consultat el 2016 des de http://db.cs.berkeley.edu/jmh/papers/cleaning-unece.pdf

Remington, Alex (8/5/2016) Understanding data journalism: Overview of resources, tools and topics. *Journalists resouce,* consultat el 2016 a http://journalistsresource.org/tip-sheets/research/understanding-data-journalism-overview-tools-topics

Dyer, Zach (17/05/2015) Journalists' beginner guide to coding. *Journalism in the Americas*, consultat el 2016 a https://knightcenter.utexas.edu/blog/00-13913-journalists'-beginner-guide-coding

The Datacamp Team (12/05/2015) Choosing R or Python for data analysis? An infographic. *Datacamp,* consultat el 2016 a https://www.datacamp.com/community/tutorials/r-or-python-for-data-analysis

Margaret Rouse (febrer de 2007). Integrated Development Environment (IDE). *TechTarget,* consultat a http://searchsoftwarequality.techtarget.com/definition/integrated-development-environment

Documentació en línia de Python, capítol 6: Modules, a https://docs.python.org/3/tutorial/modules.html

Rajaraman, A.; Ullman, J. D. (2011). "Data Mining" a Mining of Massive Datasets, pàg. 1–17. Consultat el 2016 a http://i.stanford.edu/~ullman/mmds/ch1.pdf

Moffitt, Chris (20/01/2015). Overvirew of Python Analysis Tools. *Practical business python,* consultat el 2016 a http://pbpython.com/visualization-tools-1.html

Font: Instituto nacional de estadística. “Tipos de gráficos, ¿cuál uso?”, consultat el 2016 a http://www.ine.es/explica/docs/pasos_tipos_graficos.pdf

API Rate Limits. *Twitter / Developers*, consultat el 2016 a https://dev.twitter.com/rest/public/rate-limiting

Villena Román, Julio; Luna Cobos, Adrián; González Cristbal, José Carlos (09/2014) Análisis Semántico de la opinión de los Ciudadanos en Redes Sociales en la Ciudad del Futuro, *Procesamineto del lenguaje natural, revista n.53*

De Mateo Pérez, Rosario; Bergés Saura, Laura; Sabater Casals, Marta (2009) *Gestión de empresas de comunicación.* Sevilla: Comunicación social

Annex: Codi de programació complet

1) Llista d'usuaris, nombre de tuits segons usuari i gràfic

A la primera cel·la de codi, trobem un script que llegeix el CSV i construeix una llista dels partits i el nombre de piulades que ha fet cada un, i un gràfic interactiu d'aquestes dades. A la següent, hem escrit un codi que permet esbrinar quantes piulades ha fet un usuari concret.

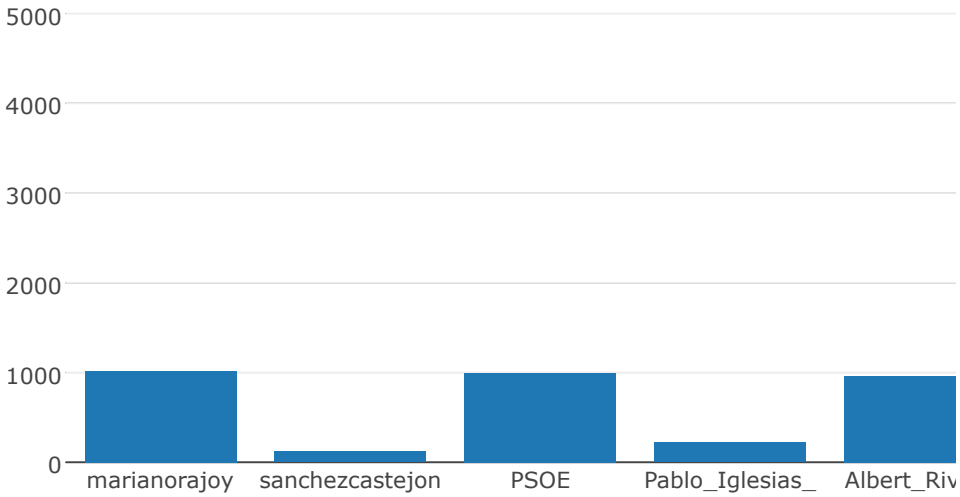
In [22]:

```
import csv
import operator
from collections import Counter

partidos = []
with open('./pol.csv') as csvfile:
    readCSV = csv.reader(csvfile, delimiter=',')
    for i, row in enumerate(readCSV):
        if i == 0:
            pass
        else:
            #linea vacia
            if row[1] == "":
                pass
            else:
                partidos.append(row[1])
partidos = dict(Counter(partidos))
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.plotly as py

data = [
    go.Bar(
        x=[str(n) for n in partidos.keys()],
        y=[str(n) for n in partidos.values()],
    )
]
print (partidos)
iplot(data)
```

```
{'marianorajoy': 1018, 'sanchezcastejon': 122, 'PSOE': 992, 'Pablo
_Iglesias_': 219, 'Albert_Rivera': 960, 'CiudadanosCs': 3620, 'aho
rapodemos': 5259, 'PPopular': 1329}
```



In [23]:

```
def cuantos_tuits(nombre):
    nb_tuits = partidos[nombre]
    print(nombre, "a tuiteado", nb_tuits, "veces")

#tests
#cuantos_tuits("marianorajoy")
#cuantos_tuits("Pablo_Iglesias_")
```

2) Qui ha dit què?

Creem una funció que troba tots els tuits de cada partit, i mostra quines han estat les paraules utilitzades amb més freqüencia

In [24]:

```
print(partidos.keys())
```

```
dict_keys(['marianorajoy', 'sanchezcastejon', 'PSOE', 'Pablo_Igles
ias_', 'Albert_Rivera', 'CiudadanosCs', 'ahorapodemos', 'PPopular'
])
```

In [25]:

```
mis_partidos = partidos.keys()
tuitsXpartido = {partido:[] for partido in mis_partidos}
print(tuitsXpartido)
```

```
{'marianorajoy': [], 'sanchezcastejon': [], 'PSOE': [], 'Pablo_Igl
esias_': [], 'ahorapodemos': [], 'PPopular': [], 'Albert_Rivera':
[], 'CiudadanosCs': []}
```

In [26]:

```
with open('./pol.csv') as csvfile:
    readCSV = csv.reader(csvfile, delimiter=',')
    for i, row in enumerate(readCSV):
        if i == 0:
            pass
        else:
            if row[1] == "":
                pass
            else:
                partido = row[1]
                tweet = row[2]
                tuitsXpartido[partido].append(tweet)


def que_tuits(nombre):
    print(nombre,"ha dicho:")
    return tuitsXpartido[nombre]

#tests
#que_tuits("PSOE")
```

Diccionari

Creem una funció que transforma majúscules en minúscules i discrimina tan paraules buides com les del nostre diccionari personalitzat de paraules que no ens interessen

In [27]:

```
from stop_words import get_stop_words
from stop_words import safe_get_stop_words
from stop_words import AVAILABLE_LANGUAGES
import string

stop_tris = ['cataluña', 'rt', 't', 'https', 'co', 'españa', 'albert_rivera','
marianorajoy','catalán','pablo_iglesias_','podemos','psoe','catalunya','ierrej
on','referéndum','va', 'españoles']
STOPWORDS = get_stop_words('spanish') + stop_tris

def get_tag_list(tweet):
    import re
    words = re.findall(r'\w+', tweet)
    return [term.lower() for term in words if term.lower() not in STOPWORDS an
d len(term) > 3]
```

3) Llista de paraules clau

Creem primer un codi que permeti comptar les paraules filtrades amb el criteri que hem descrit a l'anterior cel·la

In [28]:

```
tags = []
with open('./pol.csv') as csvfile:
    readCSV = csv.reader(csvfile, delimiter=',')
    for i, row in enumerate(readCSV):
        if i == 0:
            pass
        else:
            #linea vacia
            if row[1] == "":
                pass
            else:
                tweet = row[2]
                lista_tags = get_tag_list(tweet)
                tags.extend(lista_tags)
print(len(tags), "paraules utilitzades a les piulades")
```

```
99726 paraules utilitzades a les piulades
```

Creem una funció que compta paraules de la llista

In [29]:

```
def count_tags(tag_list):
    freq = Counter(tag_list)
    return freq
```

Out[30]:

```
[('adacolau', 1484),
 ('xavipabloada', 1349),
 ('país', 1265),
 ('madrid', 1148),
 ('inesarrimadas', 994),
 ('l6ncampaña20d', 902),
 ('problema', 897),
 ('gente', 883),
 ('creemos', 704),
 ('partidos', 703),
 ('queremos', 680),
 ('independentistas', 662),
 ('gracias', 659),
 ('fraternidad', 653),
 ('directo', 643),
 ('iglesias8aldia', 638),
 ('caraacaral6', 622),
 ('yovotopp', 622),
 ('unidad', 604),
 ('xavierdomenechs', 586),
 ('hecho', 576),
 ('partido', 522),
 ('fuerza', 511),
 ('resolver', 506),
 ('neweconomyforum', 487)]
```

Creem una visualització de les 10 parules més emprades

In [31]:

```
top_10 = top_tags(tags, 10)

from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.plotly as py

data = [
    go.Bar(
        x=[str(n[0]) for n in top_10],
        y=[str(n[1]) for n in top_10],
    )
]
iplot(data)
```

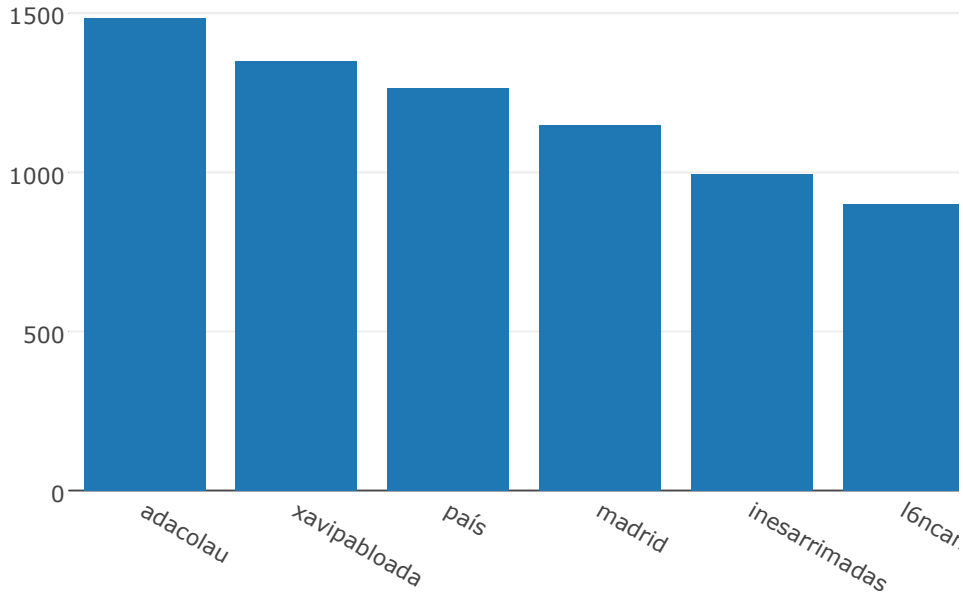
Creem una funció que torna les 25 paraules més usades independentment de l'usuari

In [30]:

```
def top_tags(tag_list, top=100):
    freq = Counter(tag_list)
    print ("Top", top, "tags:")
    for tag, nb in freq.most_common(top):
        print(tag, "ha sigut utilitzat", nb, "cops")
    return freq.most_common(top)
top_tags(tags, 25)
```

```
Top 25 tags:
adacolau ha sigut utilitzat 1484 cops
xavipabloada ha sigut utilitzat 1349 cops
país ha sigut utilitzat 1265 cops
madrid ha sigut utilitzat 1148 cops
inesarrimadas ha sigut utilitzat 994 cops
l6ncampaña20d ha sigut utilitzat 902 cops
problema ha sigut utilitzat 897 cops
gente ha sigut utilitzat 883 cops
creemos ha sigut utilitzat 704 cops
partidos ha sigut utilitzat 703 cops
queremos ha sigut utilitzat 680 cops
independentistas ha sigut utilitzat 662 cops
gracias ha sigut utilitzat 659 cops
fraternidad ha sigut utilitzat 653 cops
directo ha sigut utilitzat 643 cops
iglesias8aldia ha sigut utilitzat 638 cops
caraacaral6 ha sigut utilitzat 622 cops
yovotopp ha sigut utilitzat 622 cops
unidad ha sigut utilitzat 604 cops
xavierdomenechs ha sigut utilitzat 586 cops
hecho ha sigut utilitzat 576 cops
partido ha sigut utilitzat 522 cops
fuerza ha sigut utilitzat 511 cops
resolver ha sigut utilitzat 506 cops
neweconomyforum ha sigut utilitzat 487 cops
```

```
Top 10 tags:
adacolau ha sigut utilitzat 1484 cops
xavipabloada ha sigut utilitzat 1349 cops
país ha sigut utilitzat 1265 cops
madrid ha sigut utilitzat 1148 cops
inesarrimadas ha sigut utilitzat 994 cops
l6ncampaña20d ha sigut utilitzat 902 cops
problema ha sigut utilitzat 897 cops
gente ha sigut utilitzat 883 cops
creemos ha sigut utilitzat 704 cops
partidos ha sigut utilitzat 703 cops
```



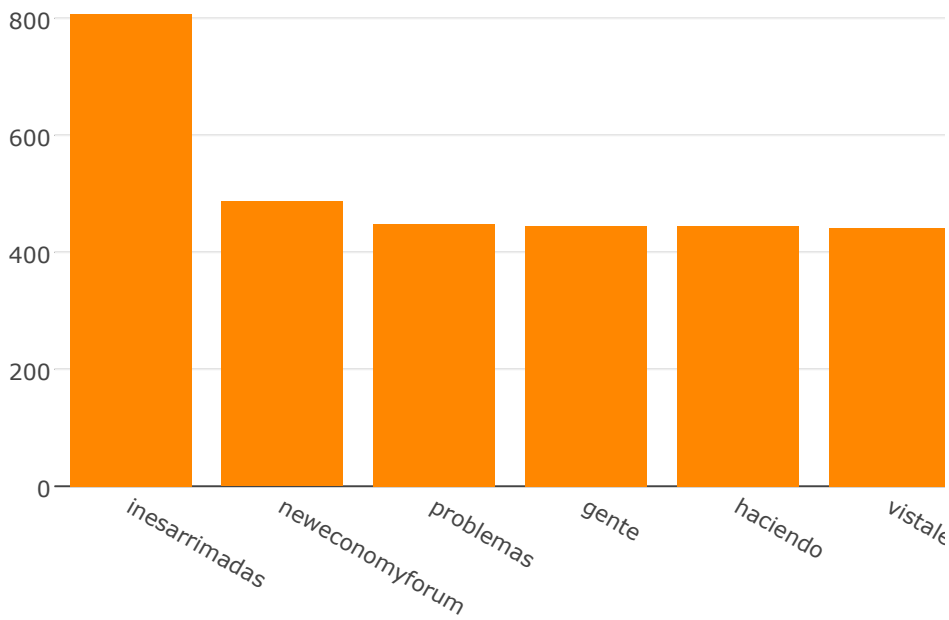
Creem un codi que troba i visualitza les 10 paraules més piulades per cada partit

In [32]:

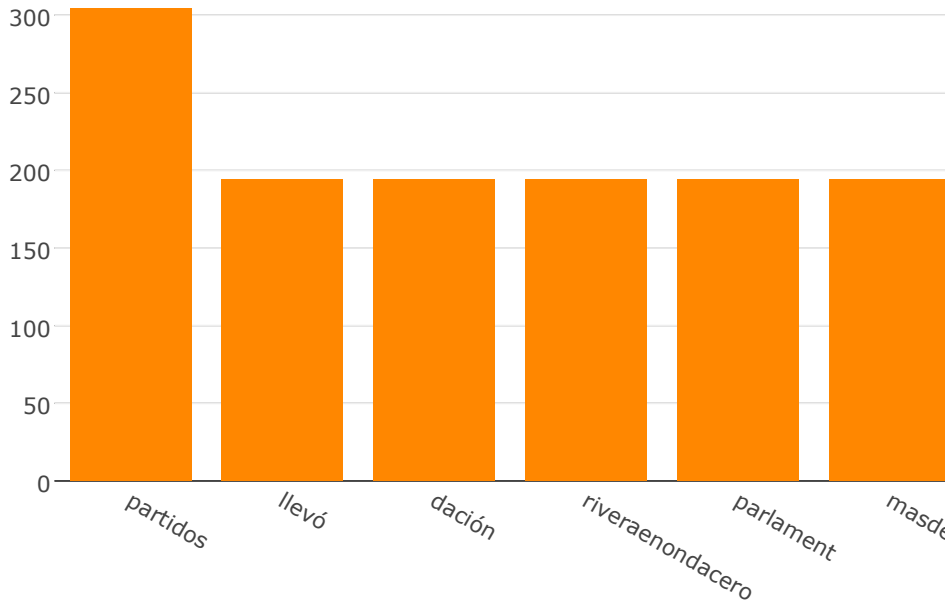
```
def top_tagsXpartidos(partido, top):
    print ("Top", top, "tags de", partido)
    tags = []
    for tuit in tuitsXpartido[partido]:
        t = get_tag_list(tuit)
        tags.extend(t)
    freq = Counter(tags)
    top_tags = freq.most_common(top)
    data = [
        go.Bar(
            x=[str(n[0]) for n in top_tags],
            y=[str(n[1]) for n in top_tags],
            marker=dict(
                color='rgb(255,135,0)',
            )
        )
        ]
    iplot(data)
    return top_tags


top_tagsXpartidos("CiudadanosCs", 10)
top_tagsXpartidos("Albert_Rivera", 10)
```

```
Top 10 tags de CiudadanosCs
```




```
Top 10 tags de Albert_Rivera
```



Out[32]:

```
[('partidos', 304),
 ('llevó', 194),
 ('dación', 194),
 ('riveraenondacero', 194),
 ('parlament', 194),
 ('masdeuno', 194),
 ('pago', 194),
 ('primer', 194),
 ('wnlsfn8h47', 194),
 ('partido', 194)]
```

4) Repartició de les paraules clau per partit i visualització

Finalment, hem descobert que el top de paraules més usades no és el mateix per cada partit o líder. Creem un codi que trobi quina és la repartició del top total entre els 8 usuaris, una visualització de gràfic de barres apilades interactiu que permet activar i desactivar les dades de paraula clau.

In [33]:

```
tagsXpartido = {k:[] for k in tuitsXpartido.keys()}
for partido, tuits in tuitsXpartido.items():
    #es una lista de tuits aqui y no un solo tuit ;)
    for tuit in tuits:
        t = get_tag_list(tuit)
        #tagsXpartido[partido] tiene una lista vacia que vamos rellenando
        tagsXpartido[partido].extend(t)
#print(tagsXpartidos['PSOE'])
```

In [34]:

```
#{"tagB":{"PSOE":X, "CC": Y, "Podemos": Z}, "tagA", {"PSOE":X, "CC": Y, "Podem
os": Z}}
repart = {}
#para cada topX tags in my tags list
for tag, freq in top_tags(tags, top=10):
    #averigo la lista de tags de cada partido
    repart[tag] = {}
    for partido, tags_list2partido in tagsXpartido.items():
        #lo ha usado? i.e esta el tag dentro de su propia lista?
        if tag in tags_list2partido:
            #aqui uso la funcion lista.count(mitag) para contar cuentas veces
en la lista apparece este tag
            repart[tag][partido] = tags_list2partido.count(tag)
        else:
            repart[tag][partido] = 0

for tag, dict_partido in repart.items():
    print(tag)
    for nombre, freq in dict_partido.items():
        print("\t", nombre, freq)
```

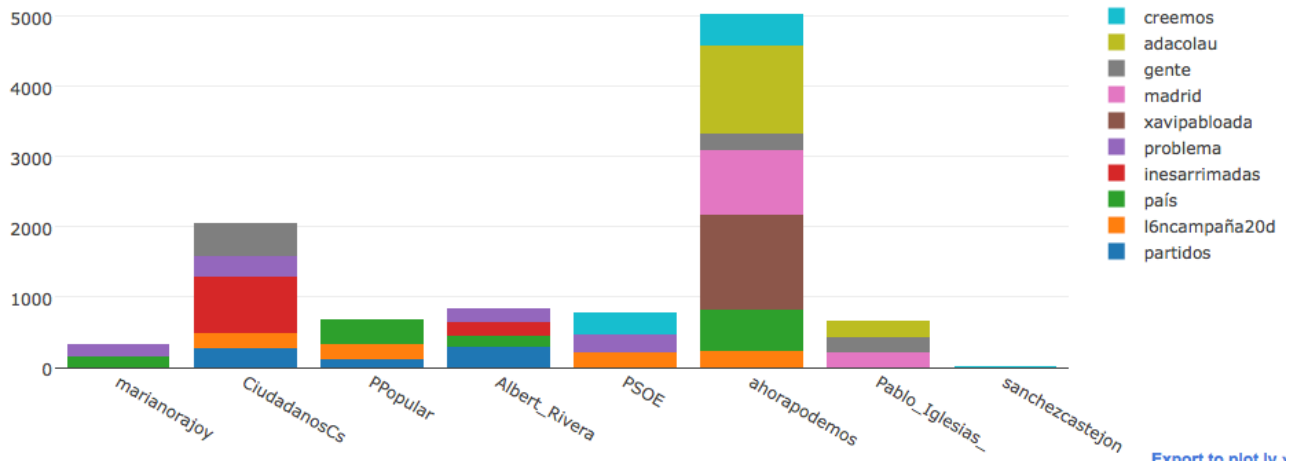
```
Top 10 tags:
adacolau ha sigut utilitzat 1484 cops
xavipabloada ha sigut utilitzat 1349 cops
país ha sigut utilitzat 1265 cops
madrid ha sigut utilitzat 1148 cops
inesarrimadas ha sigut utilitzat 994 cops
l6ncampaña20d ha sigut utilitzat 902 cops
problema ha sigut utilitzat 897 cops
gente ha sigut utilitzat 883 cops
creemos ha sigut utilitzat 704 cops
partidos ha sigut utilitzat 703 cops
xavipabloada
         marianorajoy 0
         sanchezcastejon 0
         PSOE 0
         Pablo_Iglesias_ 0
         ahorapodemos 1349
         PPopular 0
         Albert_Rivera 0
         CiudadanosCs 0
adacolau
```

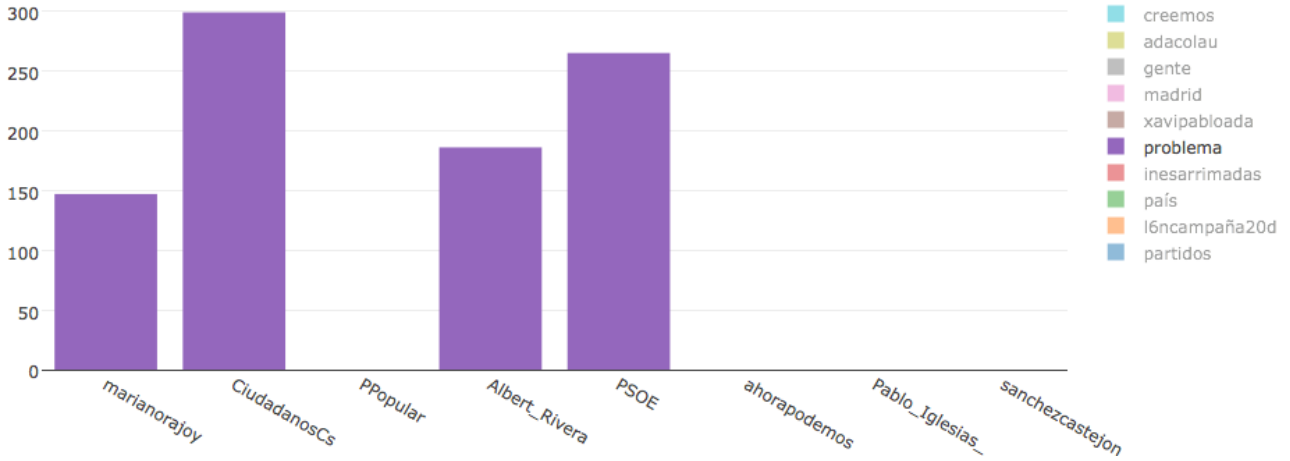
```
          marianorajoy 0
          sanchezcastejon 0
          PSOE 0
          Pablo_Iglesias_ 219
          ahorapodemos 1265
          PPopular 0
          Albert_Rivera 0
          CiudadanosCs 0
l6ncampaña20d
          marianorajoy 0
          sanchezcastejon 0
          PSOE 222
          Pablo_Iglesias_ 0
          ahorapodemos 236
          PPopular 222
          Albert_Rivera 0
          CiudadanosCs 222
país
          marianorajoy 173
          sanchezcastejon 0
          PSOE 0
          Pablo_Iglesias_ 0
          ahorapodemos 594
          PPopular 337
          Albert_Rivera 161
          CiudadanosCs 0
creemos
          marianorajoy 0
          sanchezcastejon 2
          PSOE 277
          Pablo_Iglesias_ 0
          ahorapodemos 425
          PPopular 0
          Albert_Rivera 0
          CiudadanosCs 0
gente
          marianorajoy 0
          sanchezcastejon 0
          PSOE 0
          Pablo_Iglesias_ 219
          ahorapodemos 219
          PPopular 0
          Albert_Rivera 0
          CiudadanosCs 445
problema
          marianorajoy 147
          sanchezcastejon 0
          PSOE 265
          Pablo_Iglesias_ 0
          ahorapodemos 0
          PPopular 0
          Albert_Rivera 186
          CiudadanosCs 299
partidos
          marianorajoy 0
          sanchezcastejon 0
          PSOE 0
```

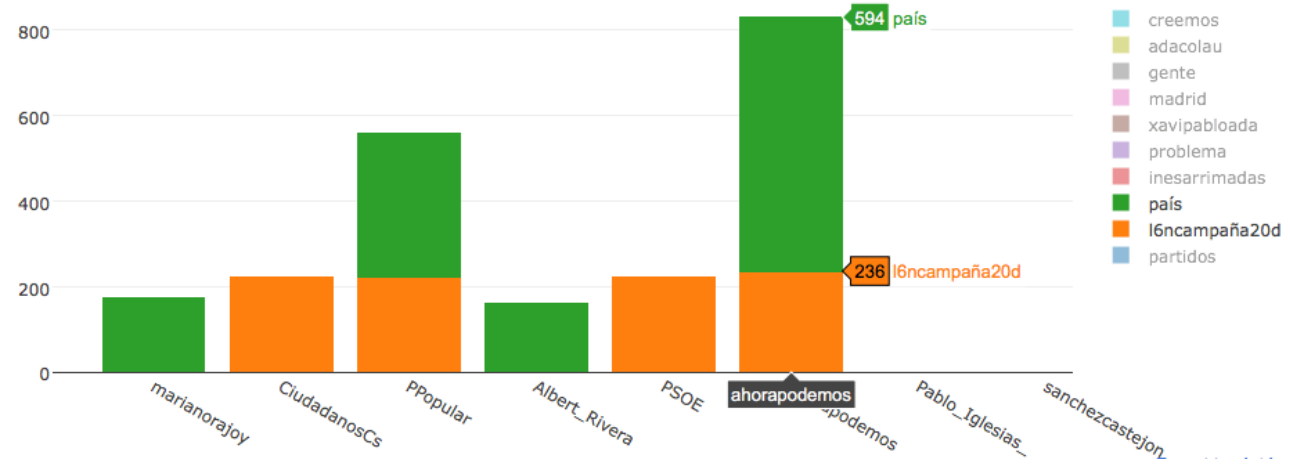
```
         Pablo_Iglesias_ 0
         ahorapodemos 0
         PPopular 125
         Albert_Rivera 304
         CiudadanosCs 274
inesarrimadas
         marianorajoy 0
         sanchezcastejon 0
         PSOE 0
         Pablo_Iglesias_ 0
         ahorapodemos 0
         PPopular 0
         Albert_Rivera 187
         CiudadanosCs 807
madrid
         marianorajoy 0
         sanchezcastejon 0
         PSOE 0
         Pablo_Iglesias_ 219
         ahorapodemos 929
         PPopular 0
         Albert_Rivera 0
         CiudadanosCs 0
```

In [19]:

```
data = []
for tag,p in repart.items():
#    print(tag, p)
    trace1 = go.Bar(
        x=[str(k) for k in p.keys()],
        y=[v for v in p.values()],
        name=tag
    )
    data.append(trace1)
layout = go.Layout(
    barmode='stack'
)
fig = go.Figure(data=data, layout=layout)
iplot(fig)
```


5000
4000
3000
2000
1000
0
marianorajoy
CiudadanosCs
PPopular
Albert_Rivera
PSOE
ahorapodemos
Pablo_Iglesias_
sanchezcastejon
creemos
adacolau
gente
madrid
xavipabloada
problema
inesarrimadas
país
l6ncampaña20d
partidos


300
250
200
150
100
50
0
marianorajoy
CiudadanosCs
PPopular
Albert_Rivera
PSOE
ahorapodemos
Pablo_Iglesias_
sanchezcastejon
creemos
adacolau
gente
madrid
xavipabloada
problema
inesarrimadas
país
l6ncampaña20d
partidos


800
600
400
200
0
594 país
236 l6ncampaña20d
marianorajoy
CiudadanosCs
PPopular
Albert_Rivera
PSOE
ahorapodemos
Pablo_Iglesias_
sanchezcastejon
creemos
adacolau
gente
madrid
xavipabloada
problema
inesarrimadas
país
l6ncampaña20d
partidos