



Universidad
Internacional
de Valencia

MÁSTER EN BIG DATA Y DATA SCIENCE

01MBID Fundamentos de la tecnología Big Data

CURSO 2021-2022

ACTIVIDAD 1: MongoDB + Twitter

Hecho por el estudiante Carlos de la Morena Coco

Configuración:

Para esta actividad, y con el fin de ahorrarme problemas con el workspace de Amazon, he decidido usar todo en local.

En mi ordenador uso el sistema operativo ubuntu 20.04, donde programo en python con la IDE de Jetbrains, PyCharm.

En lo que a configuración para la actividad se refiere, sencillamente instalé mongoDB y mongo Charts en local. Las consultas las realizo simplemente en la terminal de ubuntu.

Para instalar mongoDB en linux , y según la documentación de mongo, se siguen dos pasos:

-Añadimos el repositorio con el comando

```
wget -qO - https://www.mongodb.org/static/pgp/server-5.0.asc | sudo apt-key add -
```

-Instalamos la última versión estable de mongo

```
sudo apt-get install -y mongodb-org
```

Una vez instalado, creamos un directorio en `/data/db` para que se almacenen los datos.

La instalación de MongoDB Charts en local no está bien documentada y da lugar a innumerables errores, pero por suerte hay trabajos de la comunidad open source en github, que no requieren de instalación de docker aparte.

Yo usé el siguiente proyecto: <https://github.com/CrusaderX/mongodb-charts>

Como especifica el autor, lo único que tenemos que hacer para tener Mongo DB Charts en local es ir al directorio de la instalación y ejecutar el siguiente comando:

```
sudo docker-compose up -d
```

Habiendo parado previamente el servidor MongoDB local.

Adjunto captura de pantalla del levantamiento de este servidor en local.

```

mentefria@vivobook ~ cd Documents/charts/mongodb-charts
mentefria@vivobook ~/Documents/charts/mongodb-charts $ master$ sudo service mongodb status
● mongodb.service - An object/document-oriented database
  Loaded: loaded (/lib/systemd/system/mongodb.service; enabled; vendor prese>
  Active: active (running) since Sat 2021-11-13 10:53:28 CET; 22min ago
    Docs: man:mongod(1)
   Main PID: 1159 (mongod)
      Tasks: 23 (limit: 18968)
     Memory: 177.4M
      CGroup: /system.slice/mongodb.service
              └─1159 /usr/bin/mongod --unixSocketPrefix=/run/mongodb --config /e

ноя 13 10:53:28 vivobook systemd[1]: Started An object/document-oriented database
mentefria@vivobook ~/Documents/charts/mongodb-charts $ master$ sudo service mongodb stop
mentefria@vivobook ~/Documents/charts/mongodb-charts $ master$ ls
docker-compose.yaml README.md
mentefria@vivobook ~/Documents/charts/mongodb-charts $ master$ sudo docker-compose up -d
Starting mongo ... done
Starting charts ... done

```

Una vez hecho esto, abrimos el archivo .yaml para ver qué URL abrir para acceder desde Charts.

```

mentefria@vivobook ~/Documents/charts/mongodb-charts $ master$ gedit docker-compose.yaml
Open + docker-compose.yaml
~/Documents/charts/mongodb-charts

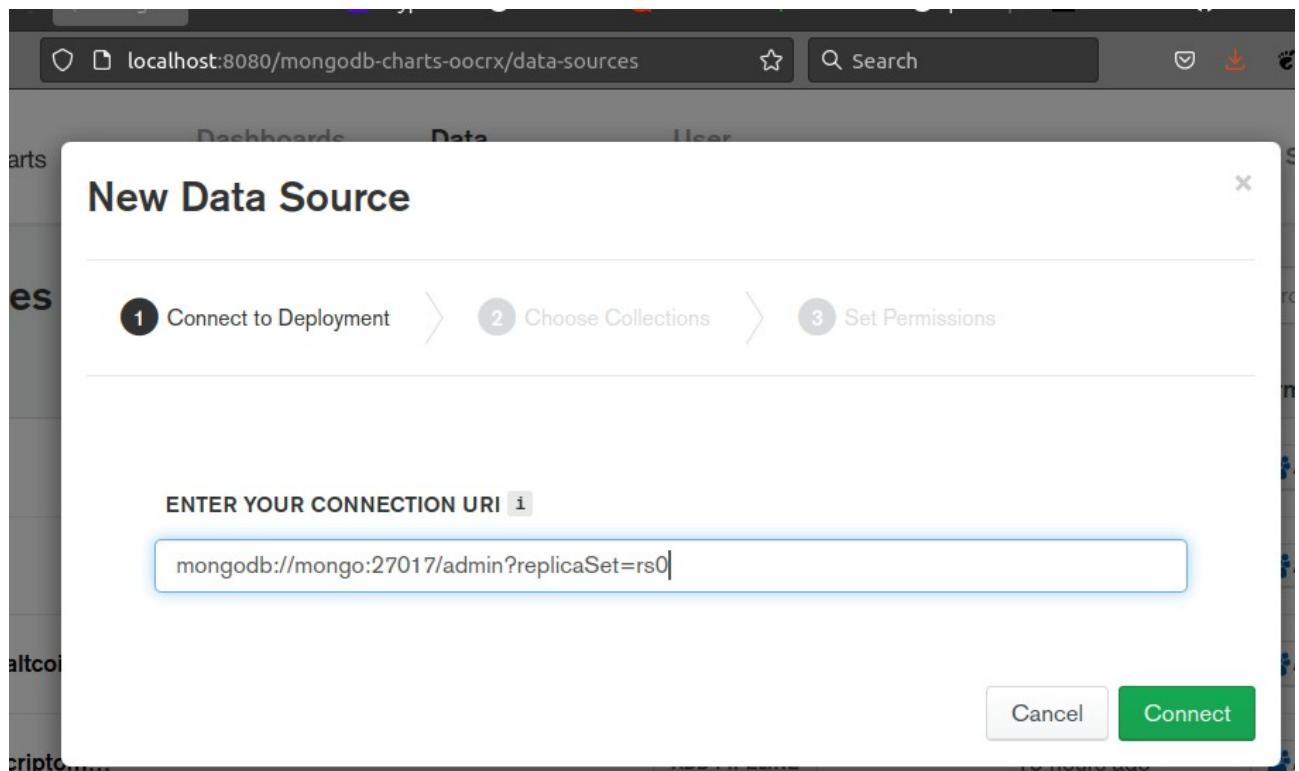
1 version: '2'
2
3 services:
4   charts:
5     build:
6       context: 'docker/charts'
7       args:
8         - EMAIL=admin@example.com
9         - PASSWORD=StrongPassw0rd
0       image: charts
1       ports:
2         - 8080:80
3       environment:
4         CHARTS_SUPPORT_WIDGET_AND_METRICS: 'on'
5         CHARTS_MONGODB_URI: 'mongodb://mongo:27017/admin?replicaSet=rs0'
6       volumes:
7         - keys:/mongodb-charts/volumes/keys
8         - logs:/mongodb-charts/volumes/logs
9         - db-certs:/mongodb-charts/volumes/db-certs
0         - web-certs:/mongodb-charts/volumes/web-certs
1       depends_on:
2         - mongo
3       container_name: charts
4
5     mongo:
6       hostname: mongo
7       build:
8         context: 'docker/mongo'

```

Aquí podemos observar el usuario y la contraseña que debemos escribir en Charts.

Para acceder a Charts, con el proyecto levantado vamos desde nuestro navegador web a localhost:8080, donde se nos abrirá automáticamente.

Una vez tengamos la base de datos creada (lo cual explicaré en el siguiente punto), todo lo que tenemos que hacer es presionar el botón de “Nueva fuente de Datos” y añadir la URL anterior, tal y como muestra la siguiente captura.



Con respecto a la base de datos, para acceder a ella desde la terminal es tan sencillo como escribir el comando “`mongo`” para acceder al servidor, y “`use dbname`” para acceder a la base de datos.

```
mentefria@vivobook ~ mongo
MongoDB shell version v3.6.8
connecting to: mongodb://127.0.0.1:27017
Implicit session: session { "id" : UUID("c7ef3bfe-5234-4e
MongoDB server version: 3.6.23
Server has startup warnings:
2021-11-13T10:16:42.393+0000 I STORAGE  [initandlisten]
2021-11-13T10:16:42.393+0000 I STORAGE  [initandlisten]
2021-11-13T10:16:42.393+0000 I STORAGE  [initandlisten]
2021-11-13T10:16:43.082+0000 I CONTROL  [initandlisten]
2021-11-13T10:16:43.082+0000 I CONTROL  [initandlisten]
2021-11-13T10:16:43.082+0000 I CONTROL  [initandlisten]
2021-11-13T10:16:43.082+0000 I CONTROL  [initandlisten]
rs0:PRIMARY> use criptomonedas
switched to db criptomonedas
rs0:PRIMARY>
```

Programa de Python:

Para crear las entradas a la base de datos, utilicé el script de python proporcionado por el profesor, con algunas modificaciones.

La primera (y más obvia) modificación fue cambiar la URL y el nombre de la base de datos y las colecciones correspondientes.

```
80     dbStringConnection = 'mongodb://localhost:27017/'  
81  
82     dbName = 'criptomonedas'  
83     dbCollectionA = 'cuentas'  
84     dbCollectionT = 'tweets'  
85
```

Lo siguiente fue añadir los campos exigidos por el profesor, oséase, añadir una entrada a cada tweet con la fecha de volcado a la base de datos y mi nombre.

```
for i in range(len(d)):  
    entry = d[i]  
    entry['estudiante_creador'] = "Carlos de la Morena Coco"  
    entry['fecha_de_volcado_a_DB'] = datetime.now()  
    entry['created_at'] = datetime.strptime(entry['created_at'], '%a %b %d %H:%M:%S +0000 %Y')
```

El lector podrá observar un tercer cambio. Esto se debe a que la fecha de los tweets obtenidos estaba en formato string, lo cual a la hora de hacer gráficos no nos permitía clasificar por fecha. Por este motivo, decidí modificar el campo “*created_at*”, para pasarlo de string a timestamp, con la biblioteca *datetime*.

Esto implica un cambio previo no mencionado, la importación de dicha biblioteca al principio del script.

```
32     import requests  
33     from datetime import datetime
```

Otro cambio notable se da en la formulación del ciclo *for*. Y es que, para cumplir con una de las exigencias de la tarea (añadir el último tweet de cada cuenta), necesitamos identificar el cuál es el primer elemento de la primera página. Para ello, hacemos el ciclo por índices. Así, sin crear ningún *bool*, ya podemos identificar de forma sencilla el primer elemento de la página, como se muestra en la siguiente captura.

```
208     try:
209         tweets.insert_one(entry)
210         if page == 1 and not i:
211             accounts.update_one({'Twitter_handle': s}, {'$set': {"earliest_tweet_in_db": entry}})
212
```

Además, esto se hace para optimizar el ciclo *while* escrito previamente. Y es que esto permite eliminar la comprobación de existencia de la página que había anteriormente. Esto ha sido otra de las modificaciones más importantes.

La comprobación de la cantidad de tweets de la página, así como el recuento de los tweets totales para este usuario y su inserción a la base de datos en este script se realiza después de la iteración e inserción de cada uno de los elementos.

Se sobreentiende que, de no haber tweets, el programa no entraría dentro del ciclo *for*.

```
220     if len(d) < 200:
221         total_tweets = (page - 1) * 200 + len(d)
222         print(f"Esta es la última página. Tweets totales de esta cuenta: {total_tweets}")
223         accounts.update_one({'Twitter_handle': s}, {'$set': {"number_of_tweets_in_db": total_tweets}})
224         break
```

En el caso de no salir del bucle *while* forzosamente anteriormente, eso quiere decir que hemos salido del bucle por cuestión del filtro de entrada, lo cual implica que se ha consultado el número indicado de páginas.

Por ello, fue añadido un bloque *else* al final del bucle *while*, el cual por definición sólo se ejecuta cuando no se ha salido forzosamente del bucle.

De esta forma hacemos la escritura del número total de tweets en el caso de tener la cuenta más tweets que nuestro máximo seleccionado.

```
249     else:
250         total_tweets = (page - 1) * 200
251         print(f"Esta es la última página. Tweets totales de esta cuenta: {total_tweets}")
252         accounts.update_one({'Twitter_handle': s}, {'$set': {"number_of_tweets_in_db": total_tweets}})
```

Consultas a la base de datos:

Mostraré cada consulta y sus resultados con una captura de pantalla, y con el comando después de cada enunciado para que sea cómodo replicar cada consulta en caso de comprobación del profesor.

Recuerdo que las consultas fueron ejecutadas directamente en la terminal de ubuntu.

-Número total de tweets.

```
db.tweets.find().count()
```

```
rs0:PRIMARY> db.tweets.find().count()
13656
rs0:PRIMARY>
```

-Número total de tweets de cada cuenta.

```
db.tweets.aggregate([{"$group": {"_id": "$user.name", "count": {"$sum": 1}}])
```

```
[{"_id": "Emin Gün Sirer", "count": 399}, {"_id": "Devchart", "count": 987}, {"_id": "BolsaZone", "count": 1154}, {"_id": "Checkmate.btc", "count": 3200}, {"_id": "David Battaglia", "count": 199}, {"_id": "CoinDesk", "count": 799}, {"_id": "PlanB", "count": 799}, {"_id": "José María Macedo", "count": 599}, {"_id": "Fred Ehrsam", "count": 1500}, {"_id": "Chris Burniske", "count": 799}, {"_id": "//Bitcoin Jack", "count": 3200}, {"_id": "Crypto.com", "count": 21}]
```

-Ránking de los 5 primeros idiomas en los que se han escrito más tweets.

```
db.tweets.aggregate([{"$group": {"_id": "$lang", "count": {"$sum": 1}}}, {"$sort": {"count": -1}}, {"$limit": 5}])
```

```
rs0:PRIMARY> db.tweets.aggregate([{"$group": {"_id": "$lang", "count": {"$sum": 1}}}, {"$sort": {"count": -1}}, {"$limit": 5}])
[{"_id": "en", "count": 11235}, {"_id": "es", "count": 1278}, {"_id": "und", "count": 850}, {"_id": "nl", "count": 112}, {"_id": "fr", "count": 37}]
```

-Ordenar cuentas de mayor a menor por número de seguidores.

```
db.tweets.aggregate([{"$group": {"_id": "$user.name", "followers": {"$first": "$user.followers_count"}}, {"$sort: {followers: -1}}])
```

```
rs0:PRIMARY> db.tweets.aggregate([{"$group": {"_id": "$user.name", "followers": {"$first": "$user.followers_count"}}, {"$sort: {followers: -1}}])  
[{"_id": "CoinDesk", "followers": 2243638 },  
 {"_id": "PlanB", "followers": 1430778 },  
 {"_id": "Crypto.com", "followers": 1140515 },  
 {"_id": "//Bitcoin Jack", "followers": 235919 },  
 {"_id": "Chris Burniske", "followers": 194826 },  
 {"_id": "Emin Gün Sirer", "followers": 191624 },  
 {"_id": "Fred Ehrsam", "followers": 170936 },  
 {"_id": "Devchart 📈", "followers": 147411 },  
 {"_id": "David Battaglia", "followers": 74588 },  
 {"_id": "Checkmate.btc 💡", "followers": 43208 },  
 {"_id": "BolsaZone", "followers": 16250 },  
 {"_id": "José María Macedo", "followers": 15632 }]
```

-20 hashtags más utilizados.

```
db.tweets.aggregate([{$unwind: "$entities.hashtags"}, {$group: {"_id": "$entities.hashtags.text", "count": {"$sum": 1}}}, {"$sort: {count: -1}}, {"$limit: 20}]]
```

```
rs0:PRIMARY> db.tweets.aggregate([{$unwind: "$entities.hashtags"}, {"$group: {"_id": "$entities.hashtags.text", "count": {"$sum": 1}}}, {"$sort: {count: -1}}, {"$limit: 20}]]  
[{"_id": "Bitcoin", "count": 366 },  
 {"_id": "bitcoin", "count": 151 },  
 {"_id": "BTC", "count": 102 },  
 {"_id": "IPO", "count": 41 },  
 {"_id": "Avalanche", "count": 37 },  
 {"_id": "Telefónica", "count": 23 },  
 {"_id": "Ethereum", "count": 16 },  
 {"_id": "SHIB", "count": 16 },  
 {"_id": "NASDAQ", "count": 15 },  
 {"_id": "Binance", "count": 12 },  
 {"_id": "NYSE", "count": 12 },  
 {"_id": "NFT", "count": 11 },  
 {"_id": "NFTs", "count": 11 },  
 {"_id": "Coinbase", "count": 9 },  
 {"_id": "Tesla", "count": 9 },  
 {"_id": "DeFi", "count": 8 },  
 {"_id": "btc", "count": 7 },  
 {"_id": "TEF", "count": 7 },  
 {"_id": "Facebook", "count": 6 },  
 {"_id": "Amazon", "count": 6 }]
```

Charts:

Para esta parte, y con el fin de que sea lo más cómodo posible para el lector, he decidido que lo mejor es guardar la captura de pantalla completa de cada chart junto con su configuración, así que tras esta página deberían seguir otras 8 páginas, cada una conteniendo un chart con su configuración, unidos directamente a este pdf sin formatear.

Si se hubiesen perdido, ruego contactarme a mi correo electrónico delamorenacoco.carlos@gmail.com

Uno de los gráficos corresponde a otra base de datos (concretamente, el del precio del bitcoin). En esta carpeta adjuntaré el archivo del cual he sacado estas cifras (bitcoin.csv).

El script que escribí para volcar estos datos a la base de datos es el siguiente:

```
1  from pymongo import MongoClient
2  import datetime
3
4
5  def sacar_info(line):
6      elementos = line.split(',')
7      fecha = datetime.datetime.strptime(elementos[1], "%Y-%m-%d")
8      simbol = elementos[2]
9      price = float(elementos[3])
10     return {"fecha": fecha, "simbolo": simbol, "precio": price}
11
12 def volcar(moneda):
13     with open(moneda) as coin:
14         objetos = [sacar_info(line) for line in coin.readlines()[2:]]
15     client = MongoClient('mongodb://localhost:27017/')
16     db = client['criptomonedas']
17     table = moneda.replace('.csv', '')
18     collection = db[table]
19     collection.insert_many(objetos)
20
21     volcar('bitcoin.csv')
22     # volcar('dogecoin.csv')
23     # volcar('ethereum.csv')
24     # volcar('verge.csv')
```

Fields
+ Add Field
filter
A id
A contributors
A coordinates
A created at
> entities { }
A estudiante creador
> extended_entities { }
favorite count
A favorited
A fecha de volcado a DB
A geo
id
A id str
A in reply to screen name
in reply to status id
A in reply to status id str
in reply to user id
A in reply to user id str
A is quote status
A lang
A place
A possibly sensitive
> quoted_status { }
quoted status id
A quoted status id str
retweet count
A retweeted
> retweeted_status { }
A source

Chart Type

Circular

Donut

Encode

Filter

Customize

Label

A name

SORT BY

VALUE

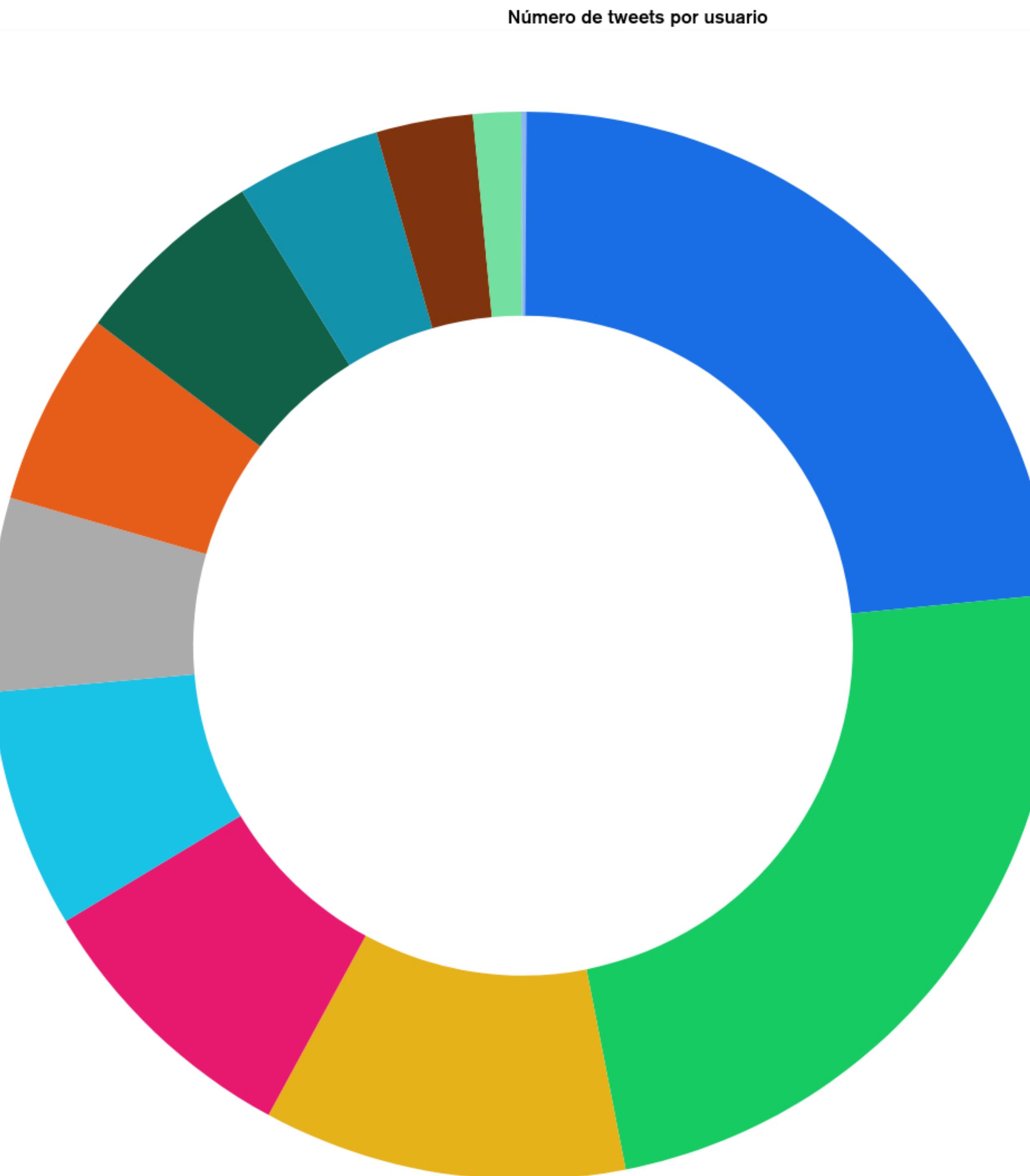
LIMIT RESULTS

Arc

id

AGGREGATE

COUNT



- Nombre de usuario
- //Bitcoin Jack
 - _Checkmate.btc
 - Fred Ehrsam
 - BolsaZone
 - Devchart
 - Chris Burniske
 - CoinDesk
 - PlanB
 - José María Macedo
 - Emin Gün Sirer
 - David Battaglia
 - Crypto.com

Fields
+ Add Field
filter
id
contributors
coordinates
created at
entities
estudiante creador
extended_entities
favorite count
favorited
fecha de volcado a DB
geo
id
id str
in reply to screen name
in reply to status id
in reply to status id str
in reply to user id
in reply to user id str
is quote status
lang
place
possibly sensitive
quoted_status
quoted status id
quoted status id str
retweet count
retweeted
retweeted_status
source

Chart Type

Bar

Grouped Stacked 100%

Show all charts

Encode Filter Customize

X Axis

text

ARRAY REDUCTIONS i

hashtags UNWIND ARRAY

AGGREGATE COUNT

+ aggregation

Y Axis

A text

ARRAY REDUCTIONS i

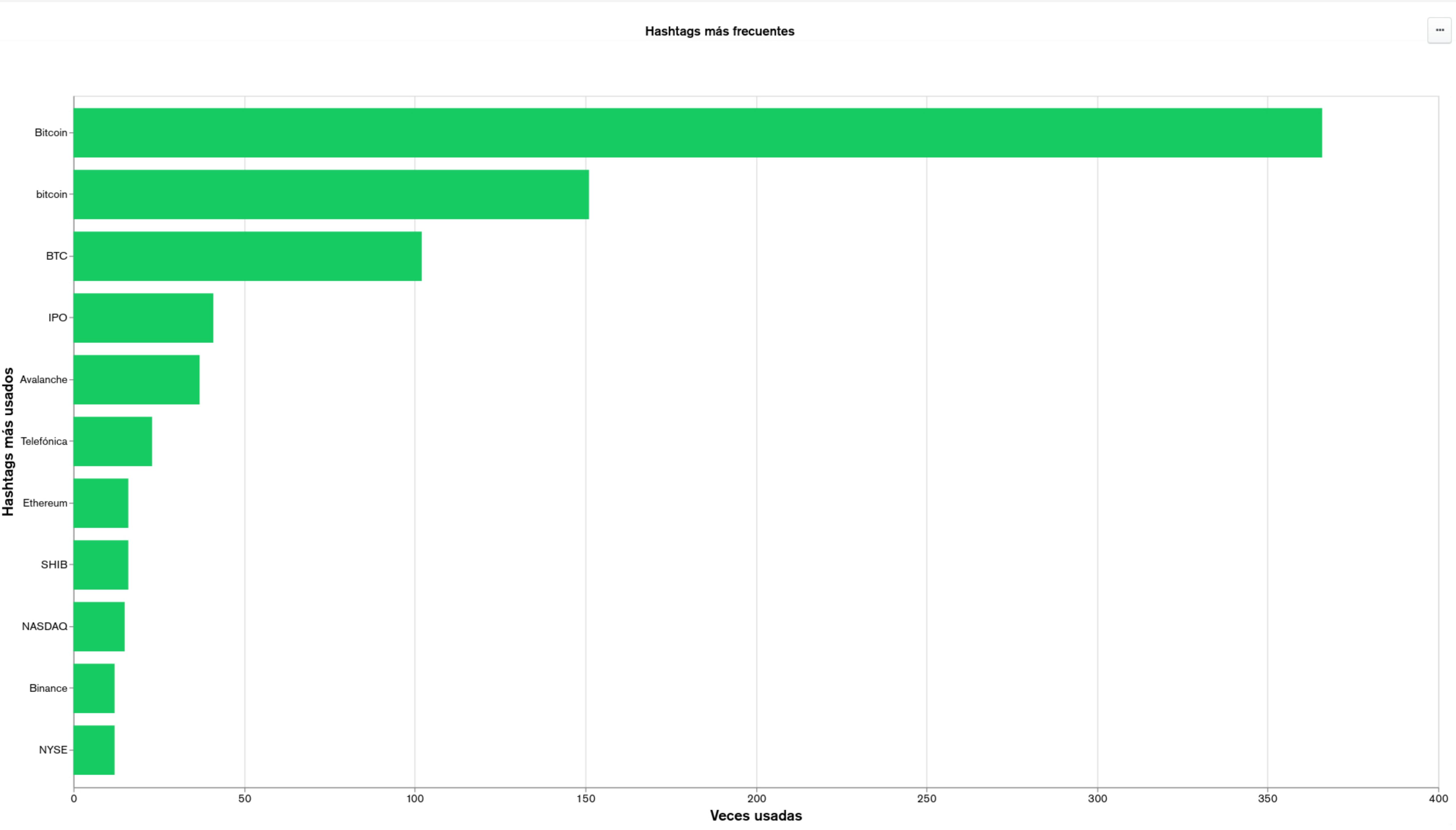
hashtags UNWIND ARRAY

SORT BY VALUE

LIMIT RESULTS SHOW: 11

Series

+ category



Fields

+ Add Field

Chart Type

filter

id

contributors

coordinates

created at

entities { }

estudiante_creador

extended_entities { }

favorite_count

favorited

fecha_de_volcado_a_DB

geo

id

id_str

in_reply_to_screen_name

in_reply_to_status_id

in_reply_to_status_id_str

in_reply_to_user_id

in_reply_to_user_id_str

is_quote_status

lang

place

possibly_sensitive

quoted_status { }

quoted_status_id

quoted_status_id_str

retweet_count

retweeted

retweeted_status { }

source

Column

Grouped

Stacked

100%

Show all charts

Encode

Filter

Customize

X Axis

 name

SORT BY

VALUE

 LIMIT RESULTS

Y Axis

 # followers count

AGGREGATE

MEAN

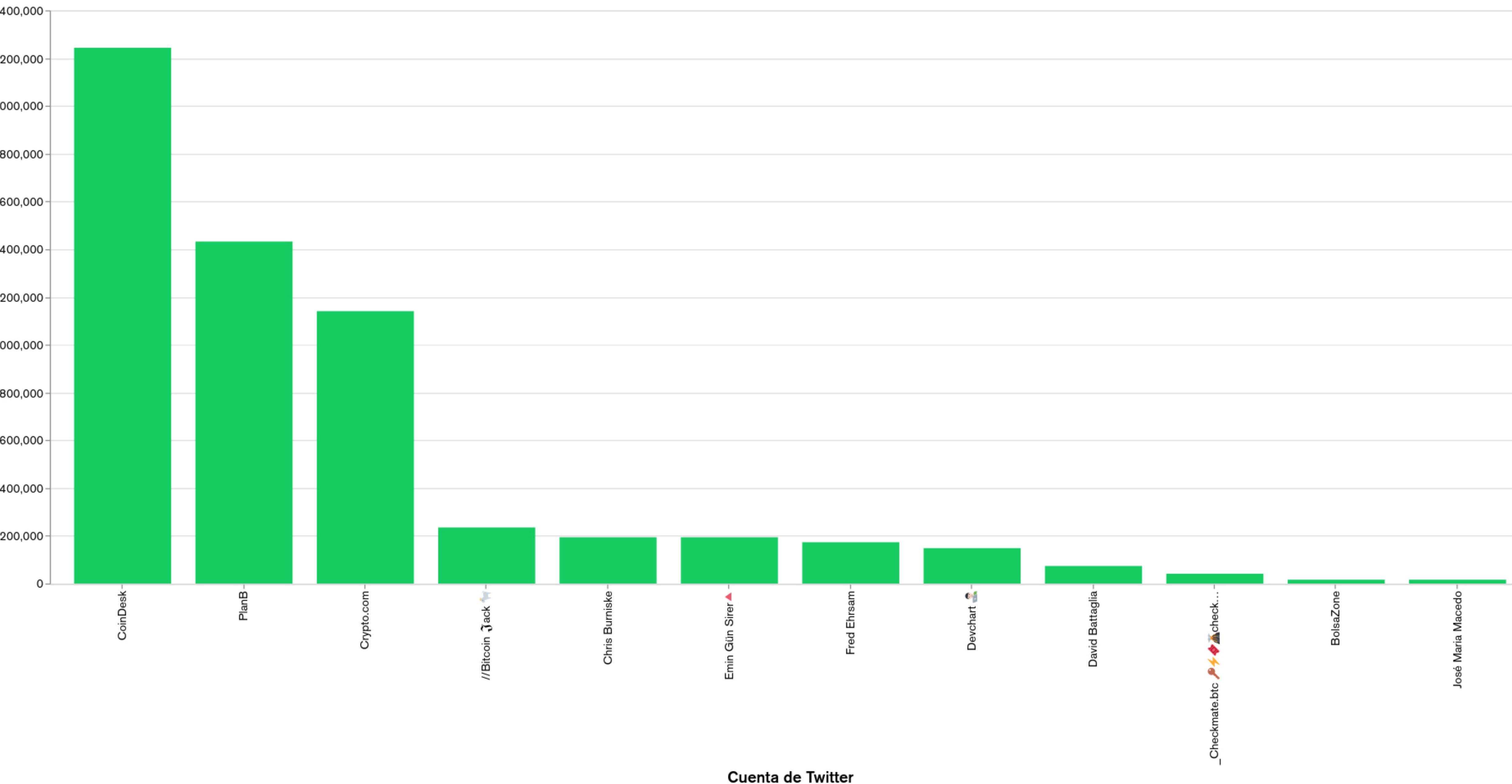
+ aggregation

Series

+ category

Seguidores de cada cuenta

Cuenta de Twitter



Fields

+ Add Field

- filter
- id
- contributors
- coordinates
- created at
- entities { }
 - estudiante_creador
 - extended_entities { }
 - favorite_count
- favorited
- fecha_de_volcado_a_DB
- geo
- id
- id_str
- in_reply_to_screen_name
- in_reply_to_status_id
- in_reply_to_status_id_str
- in_reply_to_user_id
- in_reply_to_user_id_str
- is_quote_status
- lang
- place
- possibly_sensitive
- quoted_status { }
 - quoted_status_id
 - quoted_status_id_str
- retweet_count
- retweeted
- retweeted_status { }
 - source

Chart Type

Bar

Grouped Stacked 100%

Show all charts

Encode Filter Customize

X Axis

favorite count
AGGREGATE
SUM

+ aggregation

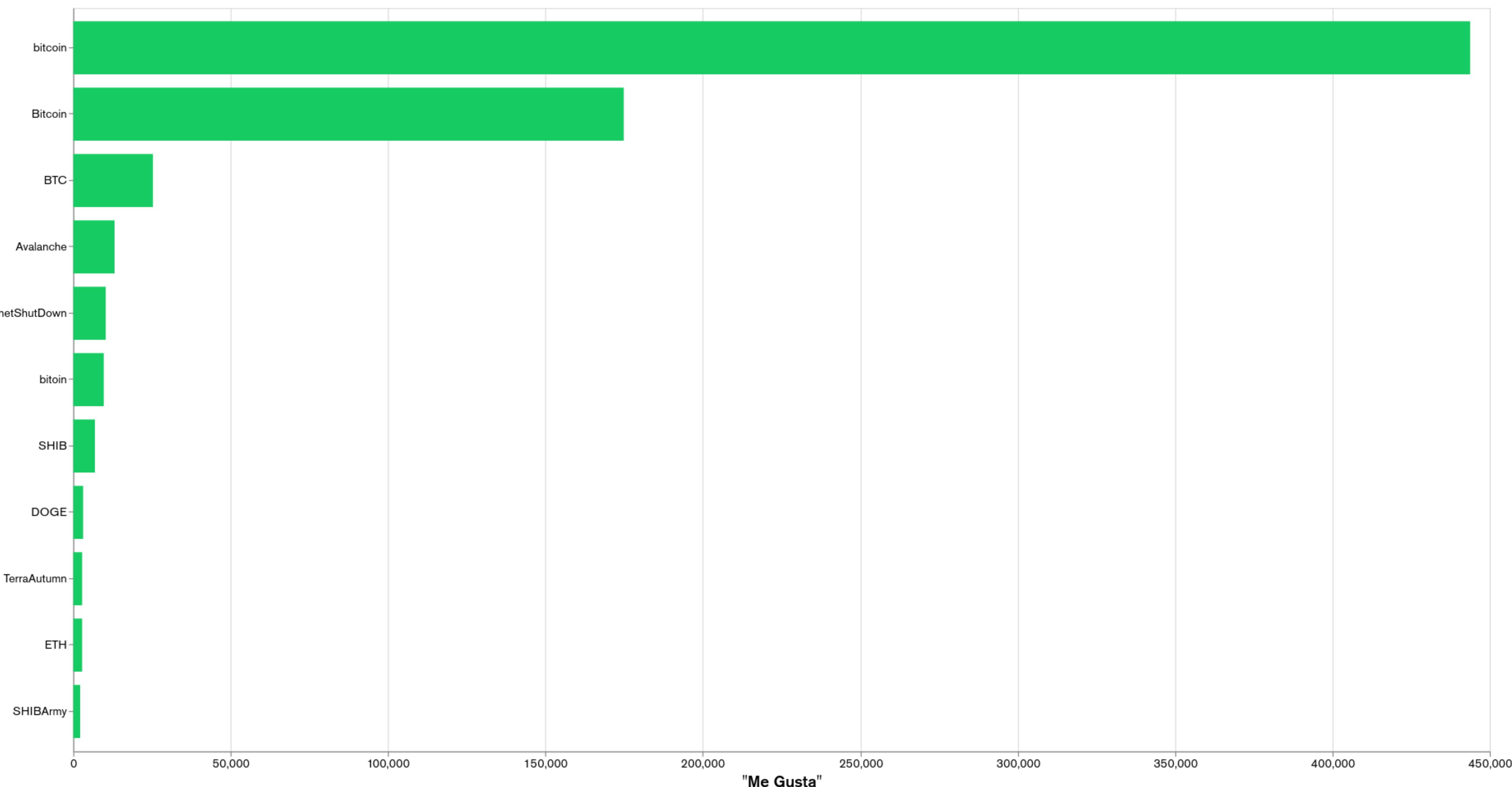
Y Axis

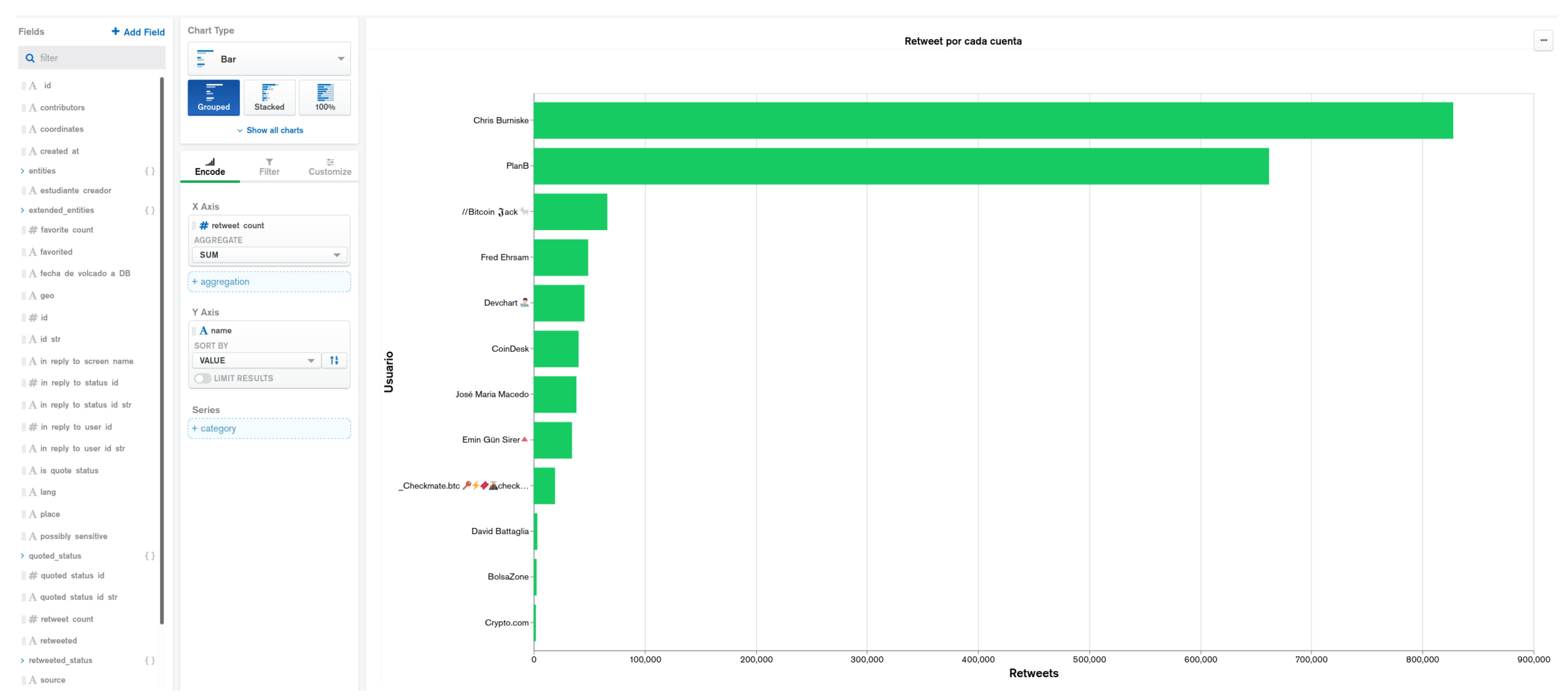
A text
ARRAY REDUCTIONS i
hashtags
UNWIND ARRAY
SORT BY
VALUE
LIMIT RESULTS
SHOW: 11

Series

+ category

Hashtags con más "Me Gusta"





Fields
+ Add Field
filter
A id
A contributors
A coordinates
A created at
> entities { }
A estudiante creador
> extended_entities { }
favorite count
A favorited
A fecha de volcado a DB
A geo
id
A id str
A in reply to screen name
in reply to status id
A in reply to status id str
in reply to user id
A in reply to user id str
A is quote status
A lang
A place
A possibly sensitive
> quoted_status { }
quoted status id
A quoted status id str
retweet count
A retweeted
> retweeted_status { }
A source

Chart Type

Circular

Donut

Encode

Filter

Customize

Label

|| A lang

SORT BY

VALUE

LIMIT RESULTS

SHOW: 10

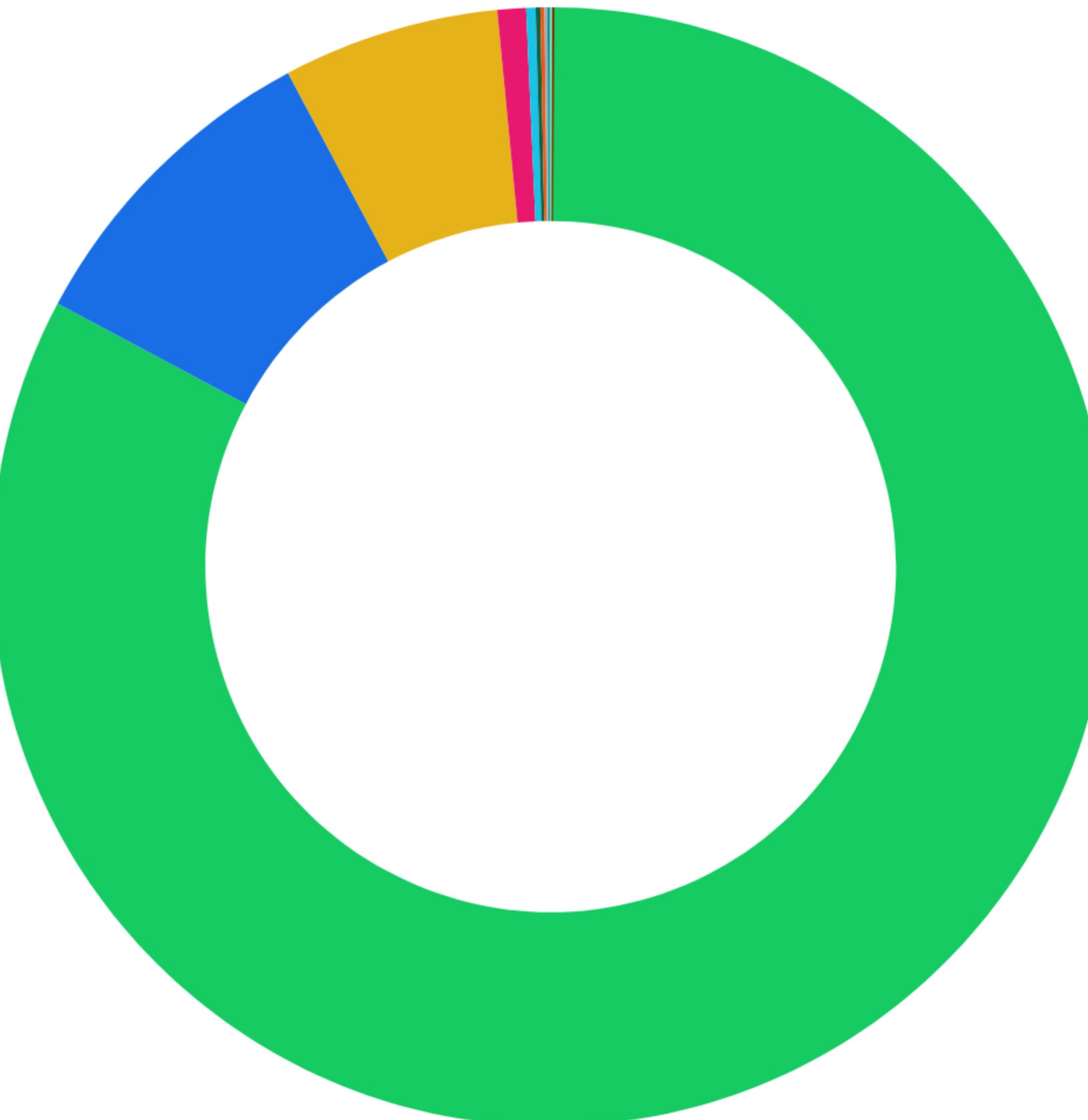
Arc

|| # id

AGGREGATE

COUNT

Idiomas en la base de datos



Idiomas

- en
- es
- und
- nl
- fr
- ht
- in
- pt
- tl
- cy
- de

Fields

+ Add Field

Chart Type

filter

Line

Discrete

Continuous

Encode

Filter

Customize

X Axis

A created at

BINNING ON

DATE OF THE MONTH

PERIODIC

Y Axis

id

AGGREGATE

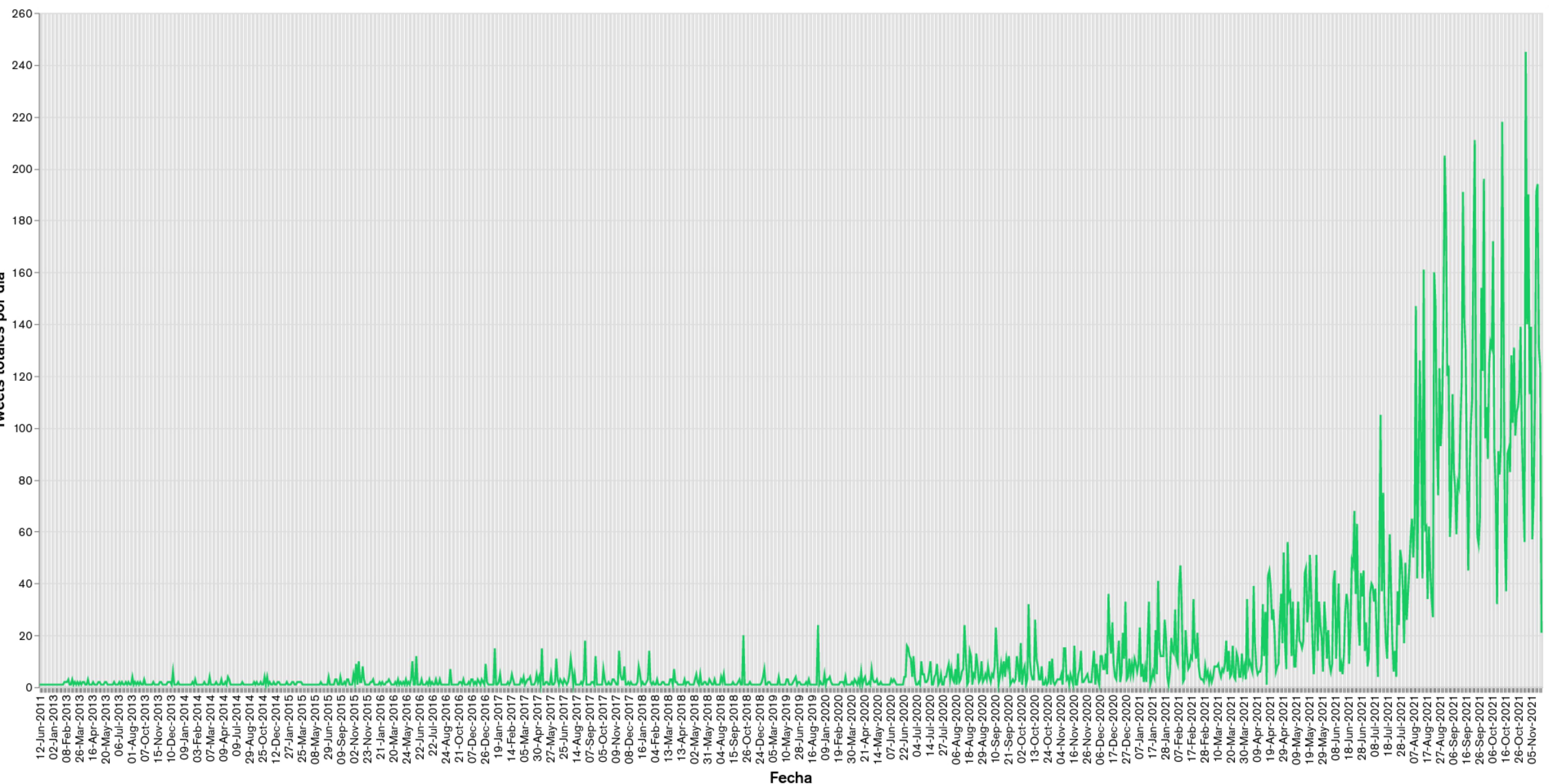
COUNT

+ aggregation

Series

+ category

Tweets for fecha



Fields

+ Add Field

Chart Type

Line

Discrete

Continuous

Encode

Filter 1

Customize

X Axis

fecha

BINNING ON

DATE OF THE MONTH

PERIODIC

Y Axis

precio

AGGREGATE

MEAN

Series

simbolo

Precios de bitcoin de 2017 a 2020

simbolo
BTCUSD