I523: Project: 020: Meetup Data Analysis and Reporting

https://gitlab.com/cloudmesh_fall2016/project-020/

Balaji Rajaram
F16-DG-4058
Indiana University
Bloomington, IN
brajaram@iu.edu

Malabika Biswas
F16-DG-4011
Indiana University
Bloomington, IN
mbiswas@iu.edu

ABSTRACT

In this paper, we present our analysis on the data collected from meetup.com[1], a social networking platform. This is an end-to-end to Big Data project that includes data consumption, pre-processing, analytics and visualization. The focus of our analysis is on two countries, the United States and India. The analysis is focused on knowing about the distribution of specific technologies across these two countries and the distribution of entrepreneurship-related activities across these countries. We explain the implementation of each of the steps in detail and provide our experimental results. This is an academic project as part of the course I523 at Indiana University. The results of the analysis are definitely surprising and provide interesting insights.

1. INTRODUCTION

The objective of the project is to analyze and visualize the meetup data to identify active cities in the meetup platform, find out the level of interest that people are showing in groups that are related to entrepreneurs, and find out trending groups in technology specifically related to Big Data technologies. These analyses may help us to get more information about the technologies that people are interested in and also it can help us to find out the locations where people are more interested in entrepreneurship. This can be helpful in carrying out experiments to find out why specific regions are showing interests in entrepreneurship and/or specific technologies and may help us to understand the driving factor for the same.

2. WHAT IS MEETUP?

Meetup[1] is a social networking platform that enables people to create groups of common interest and organize offline events for effective communication and sharing of ideas. The meetup site is organized in a hierarchical manner. The top level of the hierarchy is categories such as Arts & Culture, Books, Tech, etc. Each category is further divided into several topics such as Machine Learning, Java, PHP, etc. People interested in a specific topic can create a group of specific interest under that topic and can organize events that can be online or offline. The groups created can be made open so that anyone interested in the group can join or make it secure that requires approval from the organizer to join the group. The meetup.com also provides RESTful APIs [2] to consume the data by developers and others with certain limitations. Figure 1 shows the pictorial representation of the meetup data hierarchy.

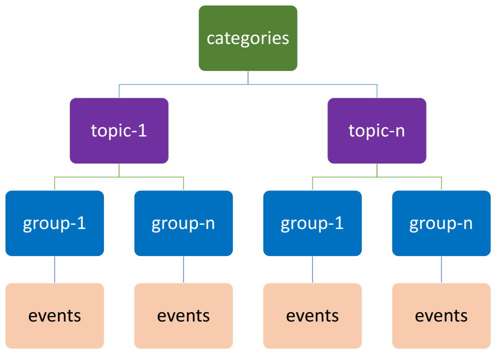


Figure 1: Meetup.com hierarchy

3. SYSTEM DESIGN

This section describes the system design of this project. We first describe the data pipeline ($3.1) and the data model ($3.2). Finally, we describe the technology stack of this project ($3.3).

3.1 Data Pipeline

This section describes the high-level end-to-end data flow of the process. The data is sourced from meetup.com that provides RESTful API for consumption. We used Python libraries to consume this RESTful API and loaded the data into HBase. The data from HBase is consumed via Spark using third party library, analytics is performed within Spark and loads the aggregated data into HDFS. We then use Python to read the data from HDFS and load it into MySQL. Tableau is used as a visualization tool which will connect to MySQL and generate reports. Figure 2 shows the pictorial representation of the data pipeline.

3.2 Data Model

This project has a Big Data NoSQL data store in HBase where the sourced data is loaded and MySQL is used to store the aggregated data. Figure 3 shows the different tables in HBase and aggregated/lookup tables in MySQL. The source is providing numerous columns but we brainstormed and identified an optimal number of columns that would suffice our reporting needs.

3.3 Technology Stack

This section describes different technologies that are involved in this project and the Hadoop cluster configuration de-

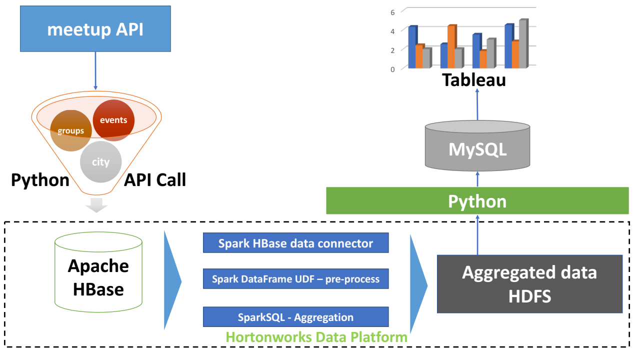


Figure 2: Meetup Data Analysis - Data Pipeline

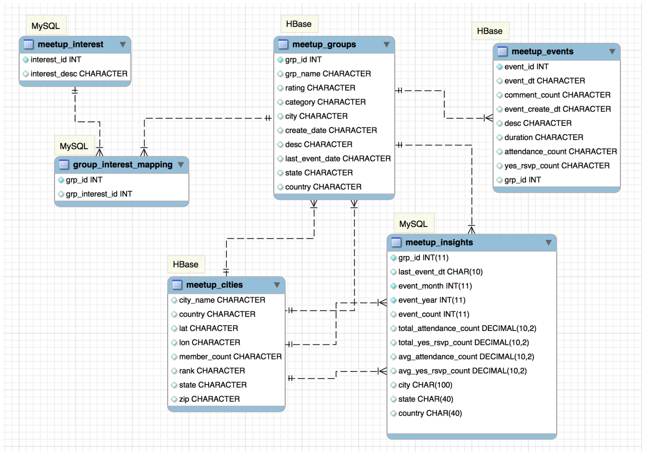


Figure 3: Meetup Data Analysis - Data Model

tails. The development was done on a three-node cluster with Hortonworks Data Platform Community Edition [3] as the Hadoop distribution. Figure 5 provides an overview of our platform. Figure 4 shows the pictorial representation of the technology stack of this project. We have used trial version of Tableau Desktop [4]. The Table 1 shows the version of software used.

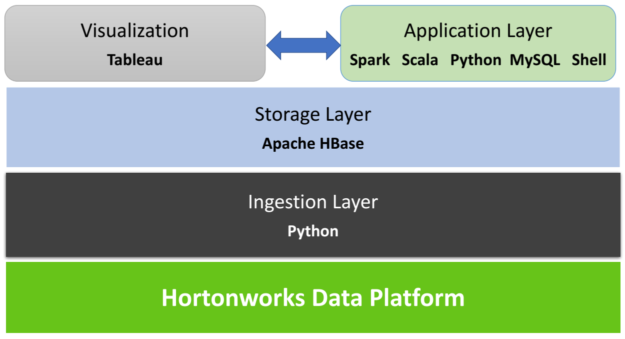


Figure 4: Technology Stack

We have used various third party libraries for this project. Below is the list of libraries used.

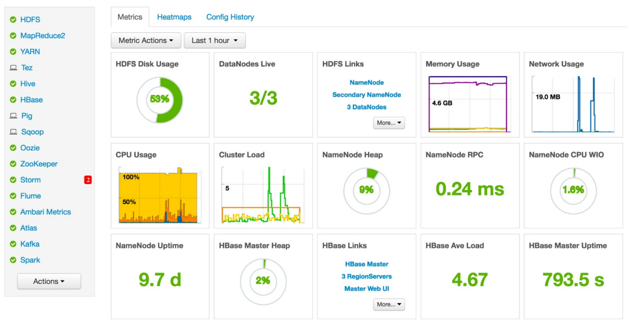


Figure 5: Hortonworks Data Platform

Table 1: Software versions

S.No	Software Name	Version
1	HDP platform	2.4.2
2	Apache Spark	1.6.1
3	Apache HBase	1.1.2
4	Python	2.7
5	Scala	2.10
6	MySQL	5.7.16
7	Tableau	10.1.1
8	Ubuntu OS	14.0.4 LTS

3.3.1 Spark-HBase connector

We are using Hortonwork's Spark-HBase connector [5] to connect to HBase from Spark. This connector supports HBase dataframe API, which was very useful during our development, testing, and interactive query analysis in the REPL. This is one of the main reasons to choose this connector over other HBase connectors and the other reason being its simplicity. We just need to create a catalog that has table and column details and then we use the standard sqlContext with the required class from the connector's library.

3.3.2 Spark-csv package

Since we were using Spark 1.6.1 version for our development, we did not have a built-in function to write CSV files with a custom delimiter. Hence, we used the spark-CSV [6] built by Databricks, which provides this functionality along with the option to specify a flag whether we need a header or not in the output file. This package is, however, built-in starting Spark 2.x.

3.3.3 Joda-Time package

The source data has datetime in epoch format and hence we had to change it to Gregorian date for which we leveraged the joda-time library [7]. We also used this library to fetch only the year and month part from the date as well.

3.3.4 TypeSafe Config Package

We wanted to parametrize the spark code without any values hard coded in it except for the column names. The tables names, lookup file name, and the output path are all parameterized and to achieve this we had to use the typesafe config package [8] which gives the functionality to load the external configuration file with variables and values in it.

Once loaded, we can use the variable in the program. This saved a lot of time during our development and testing by reducing the need to touch the code as the path and table names are configurable so we can create multiple test tables to test the program.

3.3.5 SBT

We used SBT [9] to generate the fat jar. We have included all these three third party libraries in the SBT dependency to generate the fat jar. We used the third party libraries as a comma-separated list in the packages parameter on the REPL.

3.3.6 HappyBase - Python HBase Connector

To write data into HBase from Python, we used HappyBase module, which provides a Pythonic API to interact with HBase. Below the surface, HappyBase uses the Python Thrift library to connect to HBase using its Thrift gateway.

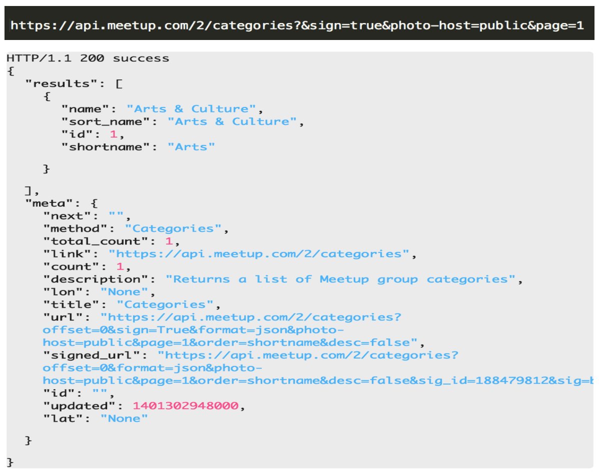
4. DATA INGESTION

This section provides an overview of different meetup APIs, explains how data has been pulled, processed and stored. It also highlights the challenges faced during data ingestion.

4.1 What is meetup API?

The Meetup API [2] provides RESTful HTTP and streaming interfaces for the developers and others with certain limitations.Typically, we can access it with a query to their server and they send back their data structured in different formats, usually in JSON format. The API is a set of core methods and a common request format. These are combined to form a URL that returns the information we want.

Figure 6 shows sample HTTP request and response for category API call.



```
https://api.meetup.com/2/categories?&sign=true&photo-host=public&page=1
```

```
HTTP/1.1 200 success
{
  "results": [
    {
      "name": "Arts & Culture",
      "sort_name": "Arts & Culture",
      "id": 1,
      "shortname": "Arts"

    }

  ],
  "meta": {
    "next": "",
    "method": "Categories",
    "total_count": 1,
    "link": "https://api.meetup.com/2/categories",
    "count": 1,
    "description": "Returns a list of Meetup group categories",
    "lon": "None",
    "title": "Categories",
    "url": "https://api.meetup.com/2/categories?
    offset=0&sign=True&format=json&photo-
    host=public&page=1&order=shortname&desc=false",
    "signed_url": "https://api.meetup.com/2/categories?
    offset=0&format=json&photo-
    host=public&page=1&order=shortname&desc=false&sig_id=188479812&sig=
    "id": "",
    "updated": 1401302948000,
    "lat": "None"

  }

}
```

Figure 6: Meetup API Request and Response

4.2 Meetup API Calls

Meetup provides a number of APIs which can be used to pull different types of data. We used Category, Cities, Groups and Events APIs. We registered in meetup.com as a user and obtained an API key, which is used to authenticate the API calls. We have used Python requests module to make API calls to fetch the data. We first pulled information about cities in the US and India, then for each city, we have fetched details about the meetup groups, and finally, we fetched details about all the events hosted for each meetup groups. Even though we pulled city details for all the cities in the US and India, we pulled groups and events details only for top cities in India and the US by population.

4.3 Process API Response

Responses have been parsed using Python JSON library. Once data is parsed, we do basic validation like null/empty check for the fields we are interested in. We then persist the validated data into a NoSQL database which is HBase.

4.4 Storing Data in HBase

We stored the results in HBase. Considering huge volume and dynamic nature of meetup data, we choose a NoSQL database HBase to store our data. HBase supports both low latency query and batch load, which has contributed to the decision of choosing HBase as the persistent data layer.

Table 2: Data Statistics

Table Name	Record Count
meetup_cities	34,592
meetup_groups	14,882
meetup_events	289,239

4.5 Data Pull Challenges

Meetup APIs are subject to a limit on how many calls can be made per second (or minute, or another short time period), in order to protect servers from being overloaded and maintain a high quality of service to many clients. The limit that is tracked and enforced over a 60-second window.

There are three important concepts here: X-Rate-Limit-Limit [10] indicates how many calls our application may make per time window. This time window is currently 60 seconds for most of our APIs, but for the most reliable code, in this case, we have uses the X-Rate-Limit-Reset header instead. X-Rate-Limit-Reset indicates when the current window ends, in seconds from the current time. X-Rate-Limit-Remaining indicates how many calls you have remaining in this window.

We have overcome this issue by monitoring X-Rate-Limit-Reset in the response header and introducing a short pause when this limit nears zero. The solution has ensured uninterrupted data flow overall.

5. DATA ANALYTICS

This section describes the analytics that was performed as part of this project. It breaks the analytic process into multiple sections. We first provide an overview of the analytics process and the list of topics that we did analysis on. We then proceed with an explanation on how the group data and interest topics are tagged and then we talk about the aggregation that was carried out on the event data.

5.1 Overview

We used Spark to carry out the analytics process. We used the spark-HBase connector to load the data into Spark HBase dataframes. We have predefined catalog for each of the tables in HBase such as groups, events, and city. We also used a predefined list of interesting topics for our analysis. We have used dataframes for standardizing and applying custom functions and we used SparkSQL for aggregations with Scala as the programming language. We have used Spark Shell for our development and analysis. We chose dataframe APIs as we were already familiar with the concept of dataframes and the another reason is that we can quickly build a table on top of the dataframes to do ad-hoc analysis using the standard SQL queries.

5.2 List of interested topics

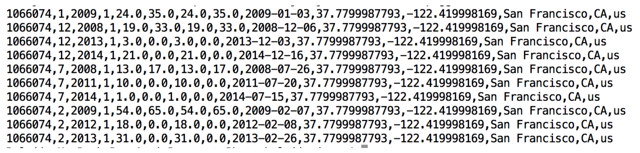
We identified around 27 trending topics for our analysis. This list is completely configurable which means that there is no need of any code change if we want to add any new topic or remove existing topic. Once the list is updated, all we need to do is, run the analytics process again and the results will be generated with the updated topic list. Following are the list of topics that we have used for our analysis: python, scala, Julia, java, r, data science, machine learning, bitcoin, artificial intelligence, blockchain, HBase, hive, spark, Hadoop, big data, solr, pig, MongoDB, cloud, elasticsearch, neo4j, smart data, fast data, internet of things,, and DevOps.

5.3 Interest - Group & Event Mapping

We have defined an integer key for each for topics that we have identified and placed it as a comma separated file in the cluster. We loaded this file into map variable in the program and broadcasted it for optimal lookup. We created a custom function which accepts the group name and group description as the parameter. It removes all nonalphanumeric characters and squeezes the spaces and concatenates the name and description to a single column. We then used the broadcasted map variable to look for the interesting topics in the concatenated column. If found, we assign the key in a comma separated variable for all the identified topics for that column. We then pivot this comma separated value into rows for each group and write it into a comma separated file using the databricks spark-CSV library. In a similar way, we identify the interest mapping for the event data and write it into a comma separated file.

5.4 Data Aggregation

We did date formatting for the event date column as the date from source is in epoch format. We have created an user-defined function to convert the date to Gregorian date format using the joda-time package. We also derived month and year columns from the processed date. We used the HBase dataframe to pull the event and city data. We then aggregated the data using SparkSQL based on group id, month and year to identify the number of events that happened for a given group, attendance count, RSVP count, and the date when the last event took place for each group as it will help to identify whether the group is still active or not. The aggregated data will be loaded into HDFS in a CSV file. Figure 7 shows the sample aggregated data. The aggregated data from HDFS is then loaded into MySQL using Python.

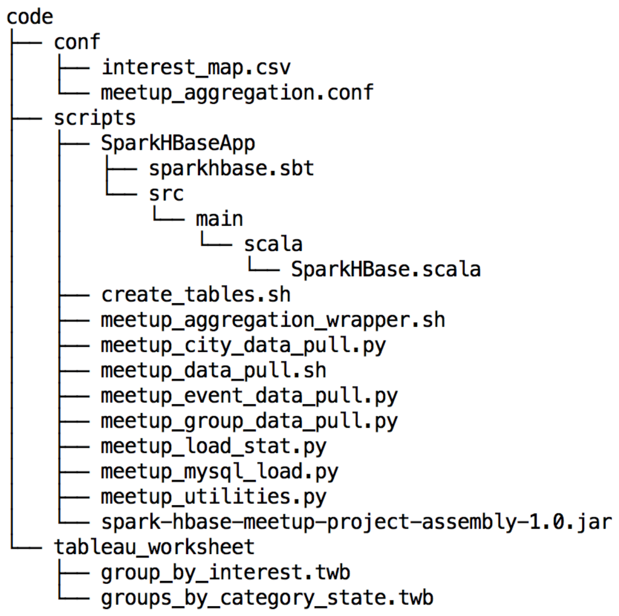


```
1066074,1,2009,1,24.0,35.0,24.0,35.0,2009-01-03,37.7799987793,-122.419998169,San Francisco,CA,us
1066074,12,2008,1,19.0,33.0,19.0,33.0,2008-12-06,37.7799987793,-122.419998169,San Francisco,CA,us
1066074,12,2013,1,3.0,0.0,3.0,0.0,2013-12-03,37.7799987793,-122.419998169,San Francisco,CA,us
1066074,12,2014,1,21.0,0.0,21.0,0.0,2014-12-16,37.7799987793,-122.419998169,San Francisco,CA,us
1066074,7,2008,1,13.0,17.0,13.0,17.0,2008-07-26,37.7799987793,-122.419998169,San Francisco,CA,us
1066074,7,2011,1,10.0,0.0,10.0,0.0,2011-07-20,37.7799987793,-122.419998169,San Francisco,CA,us
1066074,7,2014,1,1.0,0.0,1.0,0.0,2014-07-15,37.7799987793,-122.419998169,San Francisco,CA,us
1066074,2,2009,1,54.0,65.0,54.0,65.0,2009-02-07,37.7799987793,-122.419998169,San Francisco,CA,us
1066074,2,2012,1,18.0,0.0,18.0,0.0,2012-02-08,37.7799987793,-122.419998169,San Francisco,CA,us
1066074,2,2013,1,31.0,0.0,31.0,0.0,2013-02-26,37.7799987793,-122.419998169,San Francisco,CA,us
```

Figure 7: Sample Aggregated Data

5.5 Data Analytics Implementation

The entire aggregation is implemented in Spark and Scala. Automation is done using shell scripts. The fat jar is created for easier deployment and automation and parameters are configuration file driven to make it easy to move across different clusters. The configuration file has output file path, master URL, deployment mode, and HBase configuration folder path. The parameterized scripts were helpful during our testing to move between local mode and cluster mode without code change.



```
code
├── conf
│   ├── interest_map.csv
│   └── meetup_aggregation.conf
├── scripts
│   ├── SparkHBaseApp
│   │   ├── sparkhbase.sbt
│   │   └── src
│   │       └── main
│   │           └── scala
│   │               └── SparkHBase.scala
│   ├── create_tables.sh
│   ├── meetup_aggregation_wrapper.sh
│   ├── meetup_city_data_pull.py
│   ├── meetup_data_pull.sh
│   ├── meetup_event_data_pull.py
│   ├── meetup_group_data_pull.py
│   ├── meetup_load_stat.py
│   ├── meetup_mysql_load.py
│   ├── meetup_utilities.py
│   └── spark-hbase-meetup-project-assembly-1.0.jar
└── tableau_worksheet
    ├── group_by_interest.twb
    └── groups_by_category_state.twb
```

Figure 8: Project Folder Structure

6. VISUALIZATION

We used Tableau to build our visualization given the ease of installation and usability. This section describes the types of visualization and experimental results in details. We used MySQL & CSV as the source for Tableau.

6.1 Temporal Analysis

We would like to first start with our analysis on the growth of the meetup platform on India and the United States that may show the motivation behind this project. We used the derived data that has the year and event counts based on groups to perform this analysis and to plot the growth chart. The growth chart was plotted with the year on the x-axis and event counts on the y-axis. The analysis showed that the meetup platform has exponential growth over the last couple of years. The growth of this platform in India is still showing positive signs whereas the growth in the United

States is slowing down. Figure 9 shows the growth chart dashboard in Tableau that shows the overall meetup growth and the growth in India and the United States separately.

6.2 Geospatial Analysis

This section describes the geospatial analysis performed and the experimental results. The first section describes the popularity of meetup by state and the next section shows the most active cities in the meetup across the United States and India. Analysis by the state was possible only for the United States as the meetup platform provides state value only for the United States and Canada whereas it provides a null value for other countries in the world.

6.2.1 Popularity by State and Category

We used the category, state, and the group data to perform the analysis. We aggregated the number of groups by state and categories of our interest (Carrier & Business and Tech). Figure 11 shows the geographical heat map by category and state. It was interesting to note that California has more meetup groups for both the categories than any other state in the United States. Since the analysis was performed with a subset of data, we were not able to cover all the states and hence the comparison we provided is within the states for which we had data.

We have also provided state-wise analysis and popularity of tech groups stacked by interest. Figure 10 shows the stacked bar chart of this analysis.

6.2.2 Active Cities Analysis

We used the aggregated the data again to find out which cities are more active in terms of conducting events in the United States. Having more meetup groups in a city does not always mean that they are active and hence we identified the number of events conducted under each group and aggregated it by cities that shows the active cities in the meetup. Figure 13 shows the bubble chart of more active cities in the meetup.

7. ANALYSIS BY INTEREST

The analysis was performed on the group and interest mapping data. Figure 12 shows which topics have more meetup groups in the descending order. It was interesting to note that Cloud, Java, and Internet of Things are the top three topics that have more meetup groups. This shows the growth of cloud computing and Internet of Things.

8. DEVELOPMENT

We are a team of two members located in different locations and hence we used gitlab to push code updates. The project is modularized and Table 3 shows the task breakdown by us. We used Google Hangout for all the meetings including screen-sharing sessions.

9. SETUP

The project is organized as per the course requirements with code directory at the top level. Figure 8 shows how the code directory is organized. The scala source code is inside the SparkHBaseApp folder.

Table 3: Tasks Breakdown

Tasks	Participants	Communication Mode
Analysis & Design	Malabika, Balaji	Google Hangout
Data Model	Malabika, Balaji	Google Hangout
Ingestion Module	Malabika	gitlab
Analytics Module	Balaji	gitlab
Visualization Module	Malabika, Balaji	Google Hangout
report	Malabika, Balaji	Shared Sharelatex

10. FUTURE IMPROVEMENTS

We have used a simple user-defined function to identify interest topics among groups and events. We could use machine learning libraries in Spark to do a more sophisticated text analysis or by using other third party libraries through which we can get more insights. We can also use Spark streaming to pull the RSVP stream data for specific events to show the trending event topics. We could also perform analysis to find out the commonly used words in technical groups.

11. REFERENCES

[1] "Meetup," Web Page. [Online]. Available: https://www.meetup.com/

[2] "Meetup-api," Web Page. [Online]. Available: https://www.meetup.com/meetup_api/

[3] "Hortonworks data platform community edition," Web Page. [Online]. Available: http://hortonworks.com/products/data-center/hdp/

[4] "Tableau desktop trial version." [Online]. Available: https://www.tableau.com/products/desktop

[5] Z. Zhang, "Spark-on-hbase: Dataframe based hbase connector," Jun. 2016. [Online]. Available: http://hortonworks.com/blog/spark-hbase-dataframe-based-hbase-connector/

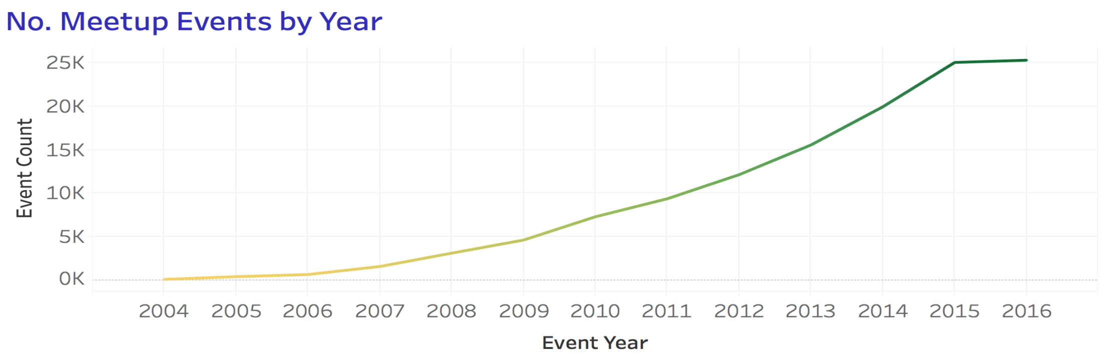
[6] "Csv data source for apache spark 1.x." [Online]. Available: https://github.com/databricks/spark-csv/blob/master/README.md

[7] "Joda-time," web. [Online]. Available: http://www.joda.org/joda-time/

[8] "Typesafe config package." [Online]. Available: https://github.com/typesafehub/config

[9] "Scala build tool," web. [Online]. Available: http://www.scala-sbt.org/0.13/docs/index.html

[10] "rate-limiting," Web Page. [Online]. Available: https://www.fullcontact.com/developer/docs/rate-limits/



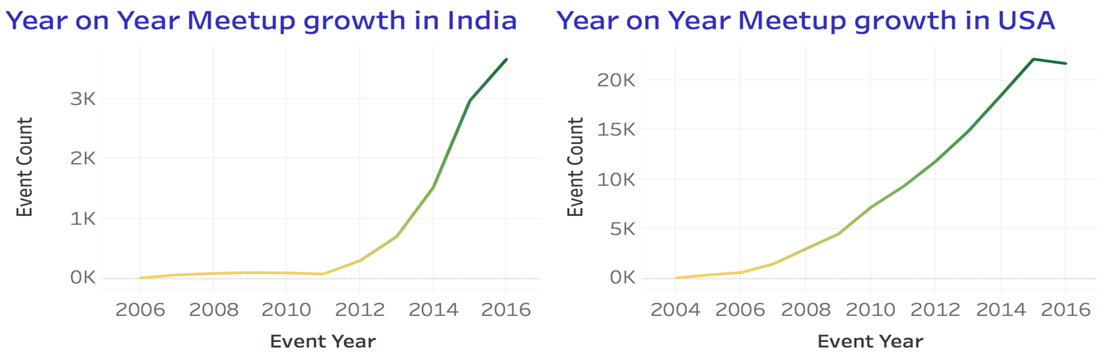


Figure 9: Meetup popularity Year on Year Growth

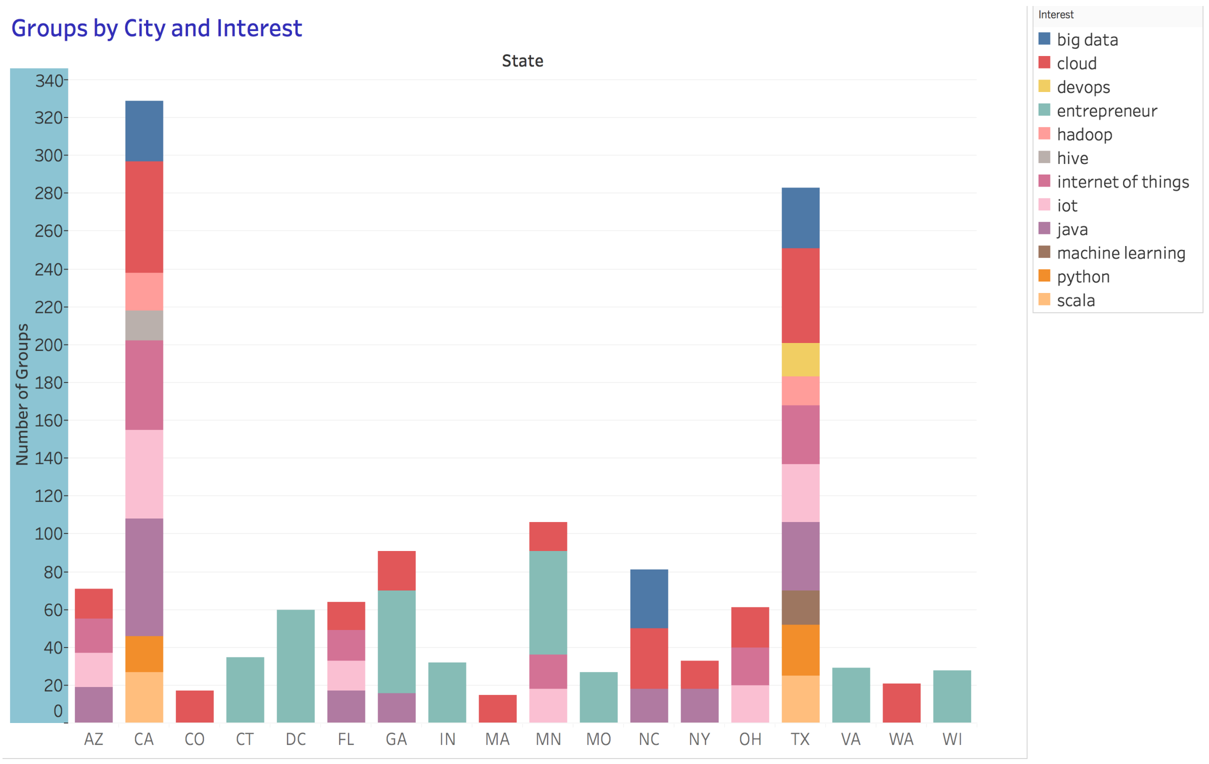


Figure 10: Groups by City and Interest

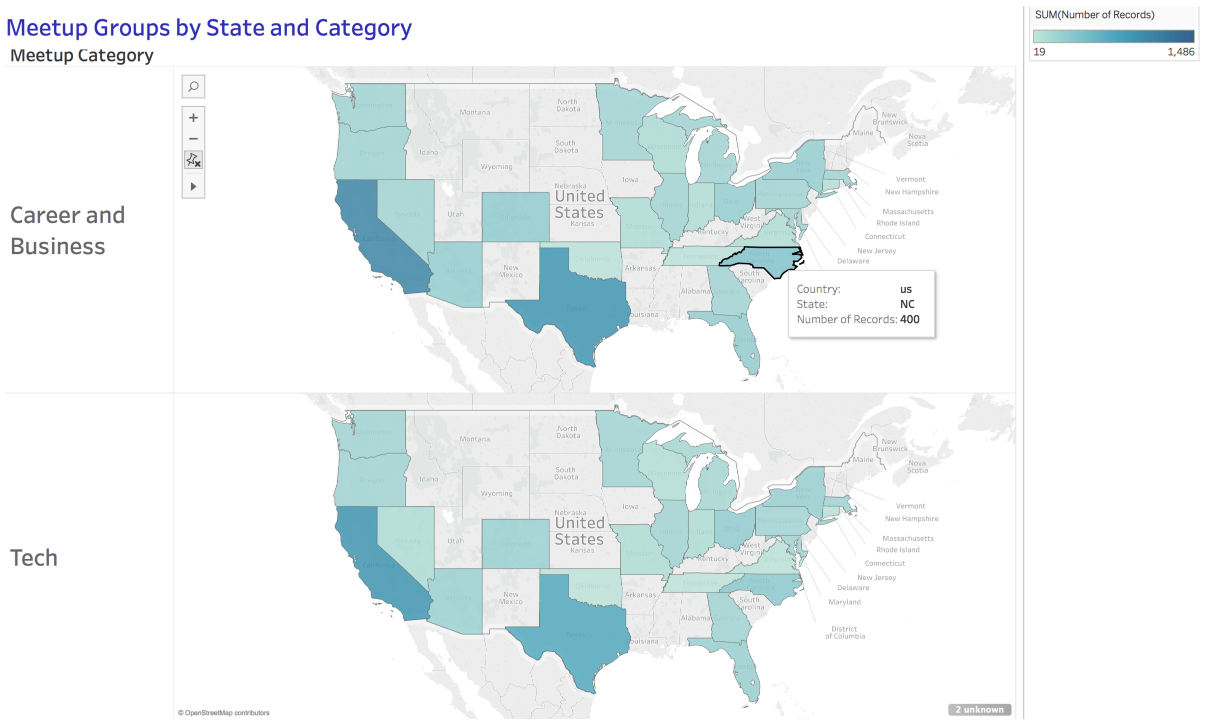


Figure 11: Groups by Category State

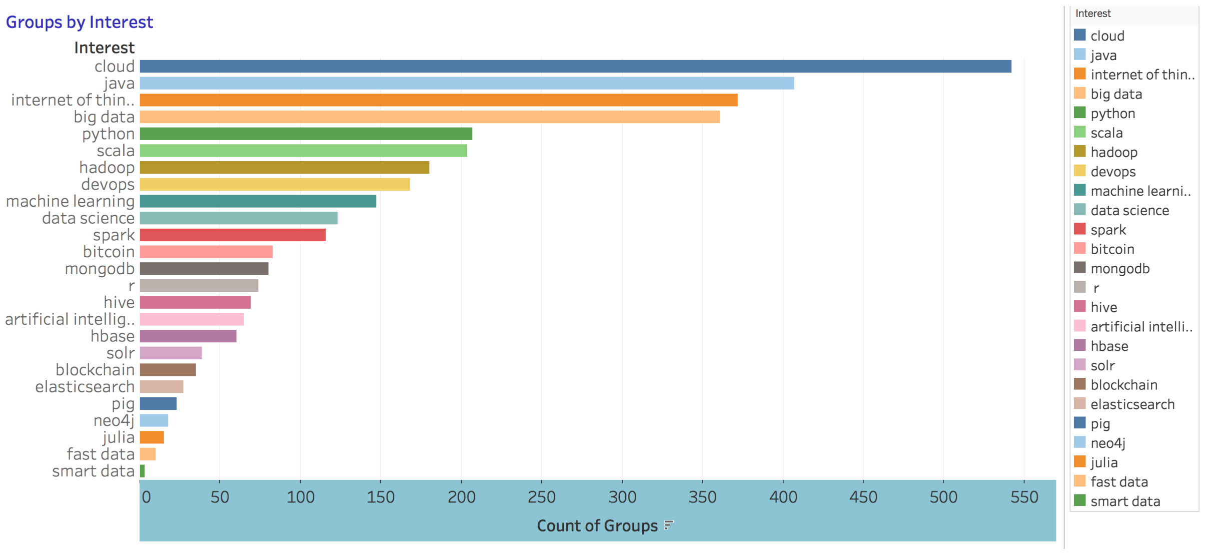


Figure 12: Groups by Interest

Active Cities by Events

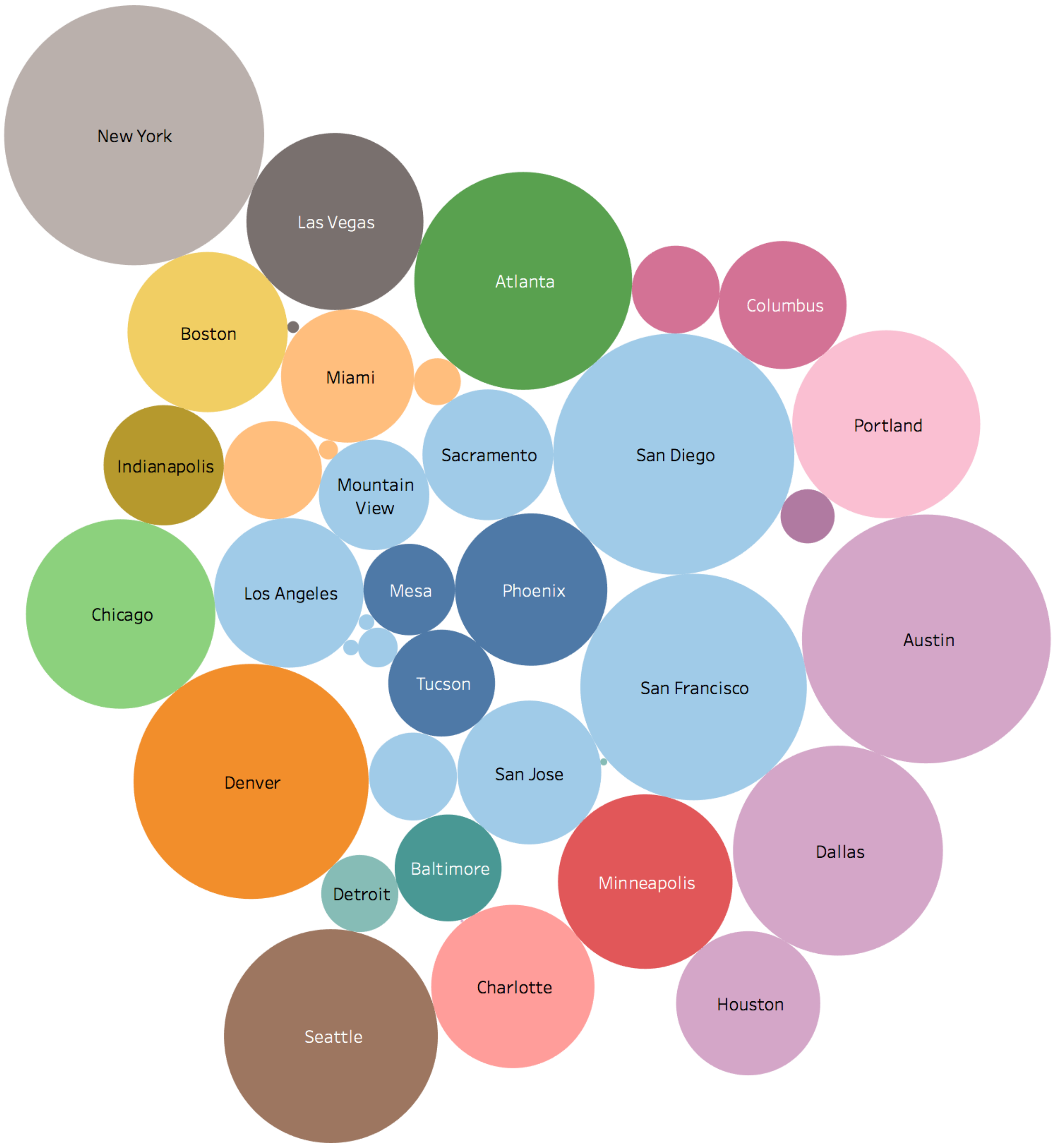


Figure 13: Active Cities by Events