

# **Personal Health Companion**

## **Technical Documentation**

### **Group 3**

AKSHITA AMBATI

HARIKA MATTA

JIANYU ZHANG

RUIQI LIN

SHIKHA KAKAR

YUEYANG CHEN

November 6, 2015

*This documentation is written to explain how the main function works from a programmer's perspective. We'll assume the programmer has a working knowledge of Java, Python, PHP, HTML, JavaScript and CSS and will only serve to show the programmer where to find and edit each portions of the code and what they do.*

## 1. Function requirements

Following are the requirements for the system-to-be:

| Identifier | Brief Description  | PW |
|------------|--|----|
| REQ1a      | The system shall attain real time tweets from twitter using stream API   | 5  |
| REQ1b      | The system shall acquire the historical health-related tweets the users sent in a certain period. (e.g. in a month or two)                           | 5  |
| REQ1c      | The system shall attain basic information from tweets of users which are health-related. (E.g. location, personal tweets numbers related, followers, | 5  |
| REQ2       | The system shall be able to access geographic information associating with Google Map.   | 4  |
| REQ3       | The system shall have a database to store whole stream of data from twitter  | 4  |
| REQ4a      | The system shall filter out irrelevant tweets in database  | 5  |
| REQ4b      | The system shall classify relevant tweets data from database by different sets. (E.g. types of exercise, locations, genders, types of foods, etc.)   | 5  |
| REQ4c      | The system shall estimate twitter users workout calories by approximate calculation.   | 3  |
| REQ5a      | The system shall provide graph to display the exercising trends.   | 3  |
| REQ5b      | The system shall provide graph to display the smoking trends.  | 3  |
| REQ5c      | The system shall provide graph to display the alcohol consumption trends.  | 3  |
| REQ5d      | The system shall provide graph to display the dietary habit trends.  | 3  |
| REQ6       | The system shall provide approximate real time tweets rolling the screen.  | 3  |
| REQ7       | The system shall do the sentiment analysis by evaluating corresponding mood state the tweets show.   | 2  |
| REQ8       | The system shall provide interactive heat map for users to check information based on geographic information   | 4  |
| REQ9       | The system shall provide dynamic hashtag cloud to show tweets users are interested in.   | 3  |
| REQ10a     | The system shall provide an approximately real time leader board on community basis.   | 2  |
| REQ10b     | The system shall be able to show top 10 ranks for work out aggregated for 3days.   | 2  |
| REQ11      | The system shall have the ability to draw information of big events correlating the themes users care about from twitter and present on a calendar.  | 2  |
| REQ12      | The system shall provide a calculator for users to show their calories loss.   | 3  |

|               |   |          |
|---------------|---|----------|
| <b>REQ13a</b> | The system shall allow user to sign up and provide an alternative option of sign up using a third party API.                | <b>5</b> |
| <b>REQ13b</b> | The system shall allow registered user to login.  | <b>5</b> |
| <b>REQ13c</b> | The system shall allow guest users to visit.  | <b>5</b> |
| <b>REQ13d</b> | The system shall allow login users to edit/delete their profile.  | <b>5</b> |
| <b>REQ13e</b> | The system shall allow user to stay login status in a moment for convenience.   | <b>1</b> |
| <b>REQ14</b>  | The system shall define authorizations for different class of users. (e.g. Registered user, Guests, Administrators)         | <b>5</b> |
| <b>REQ15</b>  | The system shall allow user to invite their friends to join the community.  | <b>2</b> |
| <b>REQ16</b>  | The system shall provide a platform (like BBS) for users to share experiences and their comments as well as making friends. | <b>2</b> |
| <b>REQ17a</b> | The system shall provide users database to store user information and retrieve data from them.                              | <b>3</b> |
| <b>REQ17b</b> | The system shall separate users according to their public information like ages and form groups of common interest.         | <b>3</b> |
| <b>REQ18a</b> | The system should be able to collect and save relevant comments.  | <b>2</b> |
| <b>REQ18b</b> | The system should be able to analyze the comments and categorize them for correlation purposes.                             | <b>2</b> |

In the above table, we list all functional requirements in three aspects as mentioned above. Some requirements divided into several parts with suffix like “a” or “b” are associated with each other.

**REQ1 to REQ3** are requirements for data collections. In this part, the system needs to attain less more relevant information from tweets and associate the data with geographical information from google map.

**REQ4** concerns all the data processing and analysis. The method we use for classification is a kind of learning machine and for data sifting part we use another algorithm for recognition of tweets’ language.

**REQ5 to REQ12** are basically our form of data output to users. As mentioned in the table, the functionalities include graph display from analyzing big data(REQ5), real time relevant tweets displaying what people are talking about(REQ6), recommendations from estimating the average level(REQ7), heat map for users to get what they concern about associating their own location(REQ8), tag cloud to show up some hashtags which users can simply select as their wish and get relevant tweets(REQ9), leaderboard displaying ranks listing with various themes and it would be additional to this system(REQ10), a calendar which provides information of big events abstracted from tweets’ data (REQ11) and a calculator for calculating calories loss in convenience(REQ12).

The remaining (**REQ13 to REQ18**) represents the way we design for users to easily get the hang of our system as well as the interactions between system and users. What's needed to be note is **REQ14** which provide our design of authorization for different classes of users. Detailed speaking, as for guest users, they can get access to data analytic graphs and tag cloud for tweets; as for login users, they are allowed to get into BBS to share information and dig deeper in data mining results so as private recommendations. They are also permitted to build their own database and save their own workout with highly security; as for administers, they have the privilege such as locking users account which are under suspension and maintaining the running of system but no access to private database.

## 2. Non-function requirement:

| Identifier     | Brief Description   | PW |
|----------------|---|----|
| <b>NF-REQ1</b> | System must be easily run with few maintenance and quickly recovered if it is broke down for some reasons.  | 5  |
| <b>NF-REQ2</b> | System must have a proper schedule for regular maintenance.   | 5  |
| <b>NF-REQ3</b> | System should entirely protect privacy issues of users and anyone should never be allowed to access others detailed information.  | 5  |
| <b>NF-REQ4</b> | System should be aesthetically designed in order to attract more users or stoppers-by.  | 5  |
| <b>NF-REQ5</b> | The system server should have the ability to handle overload in site visits.  | 5  |
| <b>NF-REQ6</b> | System should provide FAQ & efficient response for users (aka User-Friendly Design). In this part, users should be at ease when getting information and explanations about their concern. | 5  |

In the above table, **NF-REQ1** interpret the reliability of system; **NF-REQ2** shows requirements in supportability;**NF-REQ3** is most important thing for users and it applies the standard of functionality.**NF-REQ4** concerns demand for usability; **NF-REQ5** is supportability;**NF-REQ6** represents the performance. The main idea in this section is to make users willing to stay with safety and comfort and what is worth noting is that all the rules should be applied to all stages of design and be pursued carefully.

### 3. Use Cases

Use Cases Casual Description The summary of use cases is as follows:

| Use Case | Case Name                   | Description  | Requirements                              |
|----------|-----------------------------|--|---|
| UC-1     | Register                    | The system shall allow user to register, and create his profile                                    | REQ13ab<br>REQ14,REQ17a                   |
| UC-2     | View/Edit Personal Profile  | System shall display the personal profile for the logged in user.                                  | REQ13c                                    |
| UC-3     | Show Graphs                 | System shall show users graphs of several aspects(e.g. heat map, exercise time in this community ) | REQ2, REQ4ab, REQ5abcd,REQ8,REQ9, REQ10ab |
| UC-4     | Display Calendar            | System shall be able to support a calendar and allow uses to register for events.                  | REQ4ab, REQ11                             |
| UC-5     | Show Calculator             | System shall be able to support workout calculator on the home tab.                                | REQ4abc, REQ12                            |
| UC-6     | Show Leaderboard            | System shall calculate # of tweets collected per user and show the leaderboard in the UI           | REQ4ab<br>REQ10a,10b                      |
| UC-7     | Show real-time Tweets       | System shall be able to show real-time tweets with tags provided by user                           | REQ4ab, REQ6                              |
| UC-8     | A BBS section               | System shall provide users a platform for communication  | REQ16<br>REQ18ab                          |
| UC-9     | Collect Twitter Information | System shall collect and store the tweets with a given region.                                     | REQ1abc<br>REQ3, REQ4ab                   |
| UC-10    | Collect User Information    | System shall collect User Information to perform draw the necessary conclusion.                    | REQ17ab                                   |
| UC-11    | Validate User               | System shall validate when a user  | REQ14                                     |





|          |    |    |    |    |    |    |    |   |    |   |   |   |  |
|----------|----|----|----|----|----|----|----|---|----|---|---|---|--|
| REQ13b   | 5  | X  |    |    |    |    |    |   |    |   |   |   |  |
| REQ13c   | 5  | X  |    |    |    |    |    |   |    |   |   |   |  |
| REQ13d   | 5  | X  |    |    |    |    |    |   |    |   |   |   |  |
| REQ13e   | 1  |    |    |    |    |    |    |   |    |   |   |   |  |
| REQ14    | 5  | X  |    |    |    |    |    |   |    |   |   |   |  |
| REQ15    | 2  |    |    |    |    |    |    |   |    |   |   |   |  |
| REQ16    | 2  | X  |    |    |    |    |    |   |    |   |   |   |  |
| REQ17a   | 3  | X  |    |    |    |    |    |   |    |   |   |   |  |
| REQ17b   | 3  | X  |    |    |    |    |    |   |    |   |   |   |  |
| REQ18a   | 2  | X  |    |    |    |    |    |   |    |   |   |   |  |
| REQ18b   | 2  | X  |    |    |    |    |    |   |    |   |   |   |  |
| Max PW   | 5  | 5  | 4  | 2  | 3  | 2  | 5  | 2 | 5  | 3 | 5 | 4 |  |
| Total PW | 18 | 10 | 37 | 12 | 16 | 14 | 13 | 6 | 29 | 6 | 5 | 4 |  |

### 4. Use Cases Fully Dressed Description

To further develop the detailed specification of use cases, we start by drafting usage scenarios.



We thought of two possible approaches to develop use cases. One is that we first start at the data collection and another is that we start at the user functionalities like register, login, etc.

The primary goal of the system is to display data we collected to the user intuitively, but the data analytic would be fundamental to all the features. Henceforth, we decided to begin with data collection and getting the data would be the first logical step to begin data analysis too. We start from Use Case 9.

| USE CASE: UC9   | Collect Twitter Information |
|---|-----------------------------|
| <p><b>Related Requirements:</b> REQ-1abc, REQ-3, REQ-4ab</p> <p><b>Initiating Actor:</b> Administrator</p> <p><b>Actor's Goal:</b> To collect tweets from Twitter.</p> <p><b>Participating Actors:</b> Database, Twitter.</p> <p><b>Preconditions:</b> The number of API calls made should not have exceeded the limit and the administrator must be successfully authenticated by Twitter.</p> <p><b>Success End Condition:</b> The tweets are successfully collected and stored in the database.</p> <p><b>Failed End Condition:</b> The tweets are not collected from Twitter.</p> <p><b>Extension Points:</b> NA</p> <p><b>Flow of Events for Main Success Scenario:</b></p> <p>→1. Administrator makes a request to the API to get tweets. Relevant hashtags are also entered in order to get the related tweets.</p> <p>←2. Twitter sends the tweets relative to the hashtags to the administrator which is stored in the database.</p> |                             |

|   |                                 |
|---|---------------------------------|
| USE CASE: UC10  | <b>Collect User Information</b> |
| <p><b>Related Requirements:</b> REQ-17a, REQ-17b</p> <p><b>Initiating Actor:</b> Administrator</p> <p><b>Actor's Goal:</b> To collect user information extracted from the user profile from Twitter using the corresponding user id and store it in the database.</p> <p><b>Participating Actors:</b> Database, Twitter.</p> <p><b>Preconditions:</b> Successful execution of UC9; Administrator should have a valid user ID for the profile to be retrieved.</p> <p><b>Success End Condition:</b> The user information is successfully collected and stored in the database.</p> <p><b>Failed End Condition:</b> The user information is not collected from Twitter.</p> <p><b>Extension Points:</b> NA</p> <p><b>Flow of Events for Main Success Scenario:</b></p> <p>→1. Administrator searches for the user id relative to the user profile to be retrieved from the tweets received from Twitter.</p> <p>→2. Administrator sends an API request to Twitter to get the user profile. The user id is also entered in this request.</p> <p>←3. Twitter sends in the required user profiles, which are stored in the database.</p> |                                 |

The data collected through the above two use cases shall be further used in mathematical model section. The mathematical model section shall further identify

the key attributes required to develop the necessary analytics. Now, we proceed with the use cases which are defined to display our data analysis intuitively.

| USE CASE: UC3  | Show Graphs |
|--|-------------|
| <b>Related Requirements:</b> REQ-4ab, REQ-5abcd, REQ-8, REQ-9, REQ-10ab                                    |             |
| <b>Initiating Actor:</b> User  |             |
| <b>Actor's Goal:</b> To view any graph of data.  |             |
| <b>Participating Actors:</b> Database.   |             |
| <b>Preconditions:</b> Data Connectivity and instantaneity.   |             |
| <b>Success End Condition:</b> User can view any graph of data.   |             |
| <b>Failed End Condition:</b> The graph is not visible because of un-availability of required workout data. |             |
| <b>Extension Points:</b> NA  |             |
| <b>Flow of Events for Main Success Scenario:</b>   |             |
| →1. User enters the website.   |             |
| ←2. System show all graphs of data to user.  |             |
| →3. User selects location on heat map by inputting the name of location or pointing it on the map.         |             |
| ←4. System show a more specific data of chosen location to user.   |             |

| USE CASE: UC4 | Display Calendar |
|---------------|------------------|
|---------------|------------------|

**Related Requirements:** REQ-4ab, REQ-11

**Initiating Actor:** User

**Actor's Goal:** To view any upcoming event and RSVP

**Participating Actors:** Database

**Pre-Conditions:** Data Connectivity

**Success End Condition:** User can view and RSVP for events.

**Failed End Condition:** NA

**Flow of Events for Main Success Scenario:**

Include: Login (UC12)

- 1. User inputs events type he/she want to learn about.
- ← 2. System will request events information from Database.
- ← 3. System will retrieve the events information according to interests and input from users, then display the events on the calendar.
- 4. User can view the events and click on RSVP.
- ← 5. System will add the user name to the participants list in the database.
- ← 6. System will display confirmation message and mark the event as "Attending".

USE CASE: UC7

**Display Real-Time Tweets**

**Related Requirements:** REQ-4ab, REQ-6

**Initiating Actor:** User

**Actor's Goal:** To view real time tweets related to different tags

**Participating Actors:** Database

**Pre-Conditions:** Data Connectivity

**Success End Condition:** User can view interested tweets rolling the side screen.

**Failed End Condition:** NA

**Flow of Events for Main Success Scenario:**

→ 1. User inputs the tags or themes he/she want to see.

← 2. System will request relevant tweets information from Database.

← 3. System will retrieve the tweets information according to the tags input from user and display the events on the side screen.

→ 4. User can view the tweets and link to twitter.

Use Case UC8

**BBS Section**

**Related Requirements:** REQ-16, REQ-18ab

**Initiating Actor:** User

**Actor's Goal:** Use this platform to communicate with other Users.

**Participating Actors:** Database.

**Preconditions:** The database shows users' posts and comments.

**Post conditions:** Users post their opinions and comment under other users' post successfully

**Failed End Condition:** fail to post or comment

**Flow of Events for Main Success Scenario:**

Include: Login (UC12)

- 1. User clicks “Post” or “Comment” bottom
- ← 2. System display the text box.
- 3. User can edit in this section, and click “submit” to complete the process
- ← 4. System (a) stores the content to Database, (b) prompts a message “submit successfully” (c) Display the comment is the BBS section.

All these Use Cases above are concerning about how we collect data and display to users. Now we respect user to interact with our system. The interaction design can be found on some use cases listed above, i.e. BBS section has the option for login users to post some experiences, and UC4 allows login users to add their personal event to the calendar. All these interactions result in use cases designed only for users other than data. The following use cases are for users.

| Use Case UC1:  | Register |
|--|----------|
| <p><b>Related Requirements:</b>REQ-13ab, REQ-14, REQ-17a</p> <p><b>Initiating Actor:</b> User</p> <p><b>Actor’s Goal:</b> To register onto the system &amp; create a profile on system.</p> <p><b>Participating Actors:</b> Database.</p> <p><b>Preconditions:</b> (a) User must be an unregistered user. (b) System displays the homepage.</p> <p><b>Post conditions:</b> User success to register, database store the profile</p> <p><b>Failed End Condition:</b> The user have registered, and fail to register.</p> <p><b>Flow of Events for Main Success Scenario:</b></p> <p>→1. User clicks “Sign Up” option on home page.</p> <p>←2. System displays the profile page (user type (general user, governor)twitter ID, password, confirm password, gender....), and let user to fill in</p> <p>→3. User type valid ID which has not registered in the system, and type same password twice, provide all mandatory information, and click bottom “save”</p> <p>←4. System (a) prepares a database query to verify that ID has not registered in system; (b) signs to twitter to verify the twitter ID is a valid ID; (c) checks whether</p> |          |

the two password are same; (d) check all mandatory information provided

←5. Database signs the ID has not registered in the system

←6. Twitter signs the twitter ID provided is valid

← 7. System (a) stores the profile in the Database; (b) prompts a message that " Registered successfully"

### **Flow of Events for Extensions (Alternate Scenarios)**

3a. User type a valid ID which has registered in the system, and click "SAVE"

←1. System (a) detects error, (b) prompts a message "You have registered ", (c) directs to Homepage

3b. User type an invalid twitter ID, and click "SAVE"

←1. System (a) detects error, (b) prompts a message "Invalid twitter ID, please type again"

→2. User supplies a valid twitter ID

←3. Same as in Step 4

3c. User type different passwords, and click "SAVE"

←1. System (a) detects error, (b) prompts a message "Passwords do not match each other, please type again"

→2. User supplies same passwords twice

←3. Same as in Step 4

3d. User only provides part of mandatory information

←1. System (a) detects error, (b) prompts a message "You have to provide all the mandatory information, please type the rest."

→2. User supplies all the mandatory information

← 3. Same as in Step 4

3e. User click cancel option

← 1. System (a) doesn't store the profile; (b) redirects to Homepage

|  |                                   |
|--|-----------------------------------|
| Use Case UC2:  | <b>View/Edit Personal Profile</b> |
| <p><b>Related Requirements:</b> REQ-2c</p> <p><b>Initiating Actor:</b> Login User</p> <p><b>Actor's Goal:</b> To edit his/her profile information.</p> <p><b>Participating Actors:</b> Database.</p> <p><b>Preconditions:</b> The system displays the profile page</p> <p><b>Post conditions:</b> User edit his profile successfully, Database updated user's profile</p> <p><b>Failed End Condition:</b> profile failed to updated</p> <p><b>Flow of Events for Main Success Scenario:</b></p> <p>→ 1. User clicks "Edit" bottom</p> <p>← 2. System display the profile page in editing condition, and let user change</p> <p>→ 3. User type information that need be updated, and click "Save"</p> <p>← 4. System (a) stores the updated profile to Database, (b) prompts a message "updated successfully"</p> <p><b>Flow of Events for Extensions (Alternate Scenarios)</b></p> <p>3a. User type an invalid twitter ID, and click "SAVE"</p> <p>← 1. System (a) detects error, (b) prompts a message "Invalid twitter ID, please type again"</p> <p>→2. User supplies a valid twitter ID</p> <p>←3. Same as in Step 4</p> <p>3b. User type different passwords, and click "SAVE"</p> <p>←1. system (a) detects error, (b) prompts a message "Passwords do not match each other, please type again"</p> <p>→2. User supplies same passwords twice</p> <p>←3. Same as in Step 4</p> <p>3c. User click cancel option</p> <p>← 1. System (a) doesn't update the profile; (b) redirects to profile page in</p> |                                   |



cannot-editing condition































































According to the table, we mainly derive the use cases which are directly interacting with actors. These use cases might not be detailed enough, but they fully covered the features which we would like to implement.

## 5. Technical Implementation

### 1. Data Collection Part

The data is originally collected from twitter and what we are using is streaming API which is nothing fancy. After listening to the API, we formulate the data in JSON files and import them into database.

Some of the data collection snapshots are displayed here: (according to the hashtags we shortlisted)

|   |   |      |   | TweetID | TweetText   | UserID | Create Time        |   |            |                     |
|---|---|------|---|---------|---|--------|--------------------|---|------------|---------------------|
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566517672890368 | Pilateando en Olimpic !! Ya no puede entrar nadie ...                                   | 2358627272 | 2015-10-30 12:43:24 |
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566523301642240 | Pilateando en Olimpic !! Ya no puede entrar nadie ...                                   | 2358627272 | 2015-10-30 12:43:24 |
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566533946793984 | Buenas noches familiar! #SoyDeAgua #shark #surf #wa...                                  | 2358627272 | 2015-10-30 12:43:24 |
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566538640203776 | If you want to lose weight contact <a href="http://t.co/oRh...">http://t.co/oRh...</a>  | 2365562754 | 2015-10-30 12:43:24 |
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566540531838977 | Because what's a day with out my weight loss coffe...                                   | 88336290   | 2015-10-30 12:43:24 |
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566554267807744 | Entrenando con demasiado calor #GYM espalda y tric...                                   | 2756704688 | 2015-10-30 12:43:24 |
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566558722146304 | #WeightLoss #Program Health Tip: Practice a Well-B...                                   | 534928178  | 2015-10-30 12:43:24 |
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566559783313409 | #Healthy #Nutrition How to Lose Weight With Thermo...                                   | 598370918  | 2015-10-30 12:43:24 |
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566559921729536 | #EmmaWatson #Fitness #Diet ###Fitness Fitness on t...                                   | 559369958  | 2015-10-30 12:43:24 |
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566561129672704 | #FatLoss #Healthy 5 Secrets To Weight Loss While G...                                   | 601794252  | 2015-10-30 12:43:24 |
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566562564132864 | #Workout #GetFit Tips to Lose Fat <a href="http://t.co/erCn...">http://t.co/erCn...</a> | 769756746  | 2015-10-30 12:43:24 |
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566563617284096 | Después de una sesión de gym toca una duchita fres...                                   | 573865263  | 2015-10-30 12:43:24 |
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566565911179264 | #JohnMayer #Fitness Paleo Diet Success Stories htt...                                   | 784182600  | 2015-10-30 12:43:24 |
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566584429428736 | 8 Min Abs The Classic - Level 1 #workout #abs #six...                                   | 189855189  | 2015-10-30 12:43:24 |
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566588455559168 | Bem melhor assim... #Run #Mar #Floripa #VemVerão #...                                   | 98040902   | 2015-10-30 12:43:24 |
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566595267510274 | Hoy recorrimos 10 kilómetros de #running con temas...                                   | 384937373  | 2015-10-30 12:43:24 |
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566634643632129 | Before And After Weight Loss Pictures Women   <a href="http...">http...</a>             | 826488764  | 2015-10-30 12:43:24 |
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566635734159360 | RT @MisTillas: Nike LunarGlide 5 de ricardo_spv #M...                                   | 83247301   | 2015-10-30 12:43:24 |
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566644407988224 | Los Alamitos Creek Trail - equestrian route. #autu...                                   | 31481099   | 2015-10-30 12:43:24 |
| <input type="checkbox"/>  |  | Edit |  | Copy    |  | Delete | 527566658282729472 | #toronto #biking : HOW-TO: 8 tips for riding home ...                                   | 384179652  | 2015-10-30 12:43:24 |

Console

Figure: Extracted Tweets

|                  |   |           |
|------------------|---|-----------|
| 7567793953050624 | Quitting subconsciously is the first step <a href="http://t.co/0gR5MFmkEj">http://t.co/0gR5MFmkEj</a> #smoking    | 191048770 |
| 7568347810897921 | Want to Quit #Smoking? #Acupuncture Can Help You W...   | 334911759 |
| 7568535456071681 | Quitting subconsciously is the first step<br><a href="http://t.co/0gR5MFmkEj">http://t.co/0gR5MFmkEj</a> #smoking | 429023724 |
| 7568538614378496 |   | 429023724 |
| 7568710895423489 |   | 323722066 |
| 7568837924122624 |   | 82295276  |
| 7569118002950144 |   | 432136093 |
| 7569121374781440 |   | 794678975 |
| 7569153763598336 |   | 57853894  |
| 7569328087244801 | Do you love Oil Bongs?  | 58460420  |

Figure: Sample tweet depicting how people tweet about being motivated to quit smoking.

|                    |  |            |
|--------------------|--|------------|
| 527566554267807744 | Entrenando con demasiado calor #GYM espalda y tric...  | 2756704688 |
| 527566558722146304 | #WeightLoss #Program Health Tip: Practice a Well-Balanced Exercise Program <a href="http://t.co/UfsPpeVgGM">http://t.co/UfsPpeVgGM</a> #Diet | 34928178   |
| 527566559783313409 | #Healthy #Nutrition How to Lose Weight With Thermo...  | 598370918  |
| 527566559921729536 | #WeightLoss #Program Health Tip: Practice a Well-Balanced Exercise Program <a href="http://t.co/UfsPpeVgGM">http://t.co/UfsPpeVgGM</a> #Diet | 59369958   |
| 527566561129672704 |  | 01794252   |
| 527566562564132864 |  | 69756746   |
| 527566563617284096 |  | 73865263   |
| 527566565911179264 |  | 34182600   |
| 527566584429428736 |  | 39855189   |
| 527566588455559168 |  | 3040902    |
| 527566595267510274 | Hoy recorrimos 10 kilómetros de #running con temas...  | 384937373  |

Figure: Sample tweet depicting how people tweet about being motivated to lose weight.

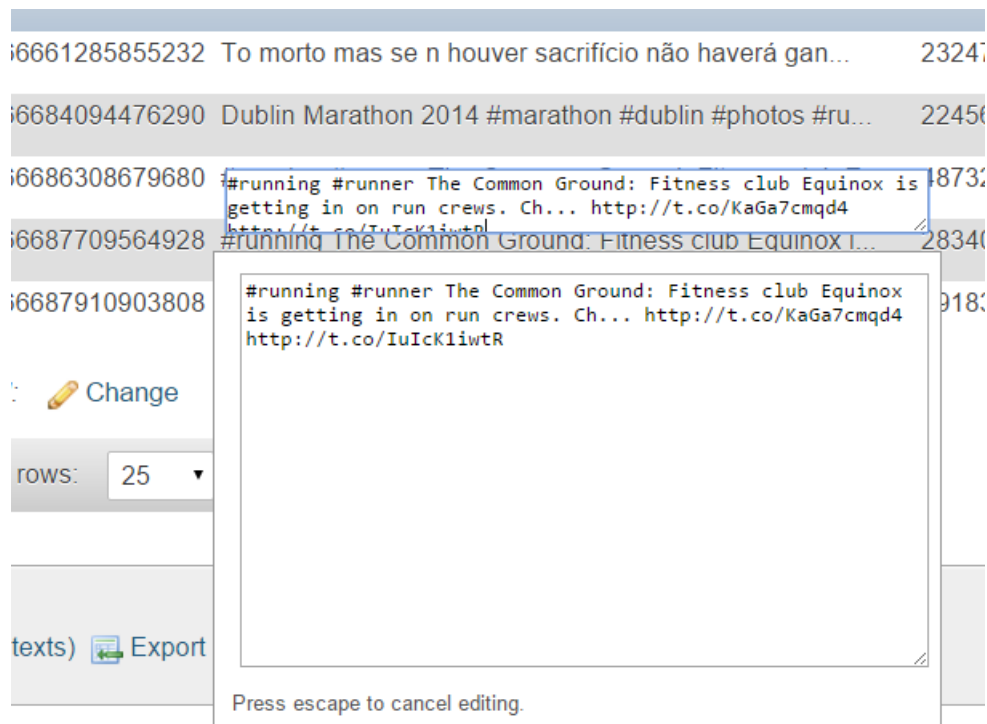


Figure: Tweet regarding fitness

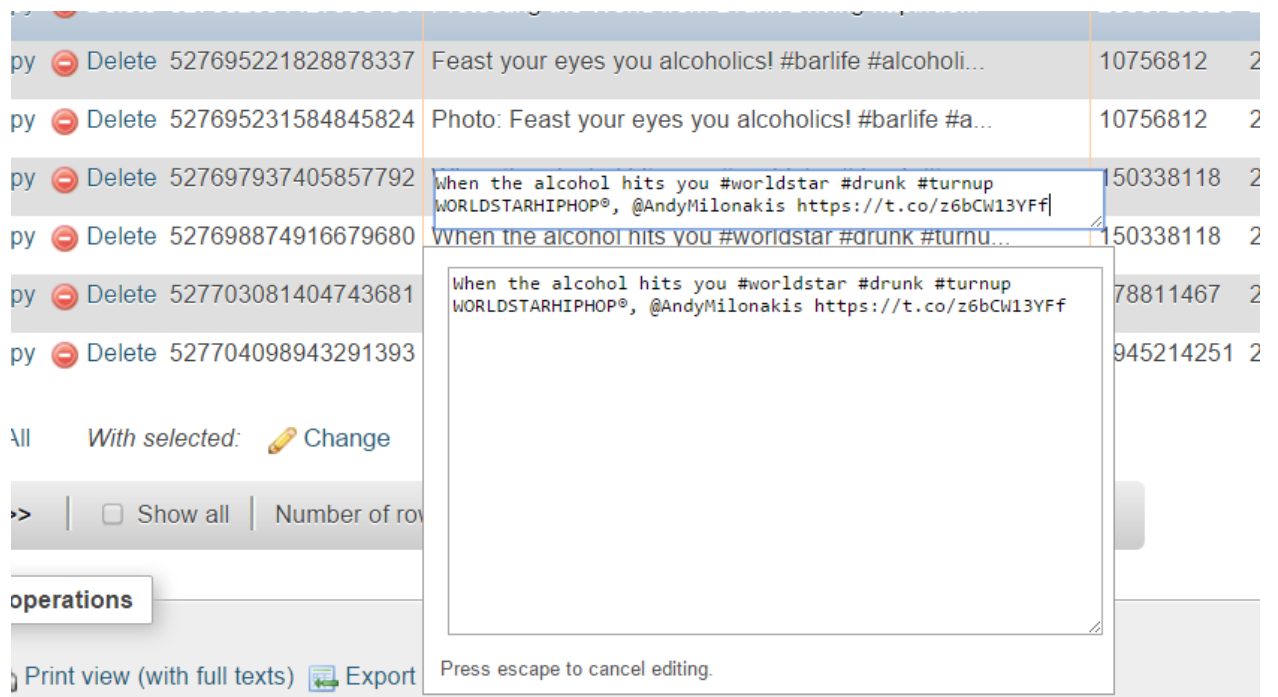


Figure: Tweet by a person who is into drinking.

## 2. Data filtering Algorithms

For the data filtering part, we use two kinds of clustering methods: Hierarchical Clustering methods and K-means Clustering method, but before use these method we have to preprocess these data (tweets).

### 2.1 Data Pre-process

We convert all data to lower case, remove punctuation, numbers, URLs, and do stemming first. After than we can make sure all data in Tweets are unique meaning words. Then use these cleaned tweets, we can get a term-document matrix which row label is the words (terms), column label is the tweet numbers (documents), and the content is the term frequency. Once we get the matrix, we can do the clustering. The sample term-document matrix (1980-2000 terms, 500-550 tweets) is as following:

| Terms            | Docs |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |  |
|------------------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|--|
|                  | 525  | 526 | 527 | 528 | 529 | 530 | 531 | 532 | 533 | 534 | 535 | 536 | 537 | 538 | 539 | 540 | 541 | 542 | 543 | 544 | 545 | 546 | 547 | 548 | 549 |  |
| smokefetishist   | 0    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| smokefree        | 0    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| smokefreegov     | 0    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| smokefreeteen    | 0    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| smokeless        | 0    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| smokers          | 0    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 2   |  |
| smokey           | 0    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| smokinfitgirls   | 0    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| smoking          | 0    | 0   | 0   | 0   | 0   | 1   | 0   | 0   | 0   | 0   | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 1   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| smokingampriding | 0    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| smokingban       | 0    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| smokingcessation | 0    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| smokingdp        | 0    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| smokingfetish    | 1    | 1   | 0   | 0   | 0   | 0   | 1   | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| smokinggirl      | 0    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| smokingicknow    | 0    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| smokingthe       | 0    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| smoko            | 0    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| snoopdogg        | 0    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| snow             | 0    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |
| snuf             | 0    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |  |

When we get the matrix, it is easy to get the frequency of terms, just the sum of the row numbers.

Then after removing sparse (we set sparsity as 0.95), we can cluster terms.

#### 2.1.1 Hierarchical Clustering

For the hierarchical clustering, we use Wald's method using a set of dissimilarities for the n objects being clustered. Initially, each object is assigned to its own cluster and then proceeds iteratively, at each stage joining the two most similar clusters, continuing until there is just a single cluster. At each stage distances between clusters are recomputed by the Lance–Williams dissimilarity update formula, which is as following:



Lance and Williams (1967) established a succinct form for the update of dissimilarities following an agglomeration. The parameters used in the update formula are dependent on the cluster criterion value. Consider clusters (including possibly singletons)  $i$  and  $j$  being agglomerated to form cluster  $i \cup j$ , and then consider redefining the dissimilarity relative to an external cluster (including again possibly a singleton),  $k$ . We have:

$$d(i \cup j, k) = a(i) \cdot d(i, k) + a(j) \cdot d(j, k) + b \cdot d(i, j) + c \cdot |d(i, k) - d(j, k)|$$

where  $d$  is the dissimilarity used – which does not have to be a Euclidean distance to start with, insofar as the Lance and Williams formula can be used as a repeatedly executed recurrence, without reference to any other or separate criterion; coefficients  $a(i), a(j), b, c$  are defined with reference to the clustering criterion used (see tables of these coefficients in Murtagh, 1985, p. 68; Jambu, 1989, p. 366); and  $|\cdot|$  denotes absolute value.

The Lance-Williams recurrence formula considers dissimilarities and not dissimilarities squared.

A number of different clustering methods are provided. *Ward's* minimum variance method aims at finding compact, spherical clusters. Two different algorithms are found in the literature for Ward clustering. We implement the criterion Murtagh (1985) and Legendre (2012).

We start with (let us term it) the Ward1 algorithm as described in Murtagh (1985).

It was initially Wishart (1969) who wrote the Ward algorithm in terms of the Lance-Williams update formula. In Wishart (1969) the Lance-Williams formula is written in terms of squared dissimilarities, in a way that is formally identical to the following.

Cluster update formula:

$$\delta(i \cup i', i'') = \frac{w_i + w_{i'}'}{w_i + w_{i'} + w_{i''}} \delta(i, i'') + \frac{w_i' + w_{i'}''}{w_i + w_{i'} + w_{i''}} \delta(i', i'') - \frac{w_{i'}''}{w_i + w_{i'} + w_{i''}} \delta(i, i')$$

$$\text{and } w_{i \cup i'} = w_i + w_{i'} \tag{3}$$

We now look at the Ward2 algorithm described in Kaufman and Rousseeuw (1990) and Legendre and Legendre (2012).

At each agglomerative step, the extra sum of squares caused by agglomerating clusters is minimized, exactly as we have seen for the Ward1 algorithm

above. We have the following.  
Cluster update formula:

$$\delta(i \cup i', i'') = \left( \frac{w_i + w_i''}{w_i + w_{i'} + w_{i''}} \delta^2(i, i'') + \frac{w_{i'} + w_i''}{w_i + w_{i'} + w_{i''}} \delta^2(i', i'') - \frac{w_i''}{w_i + w_{i'} + w_{i''}} \delta^2(i, i') \right)^{1/2}$$

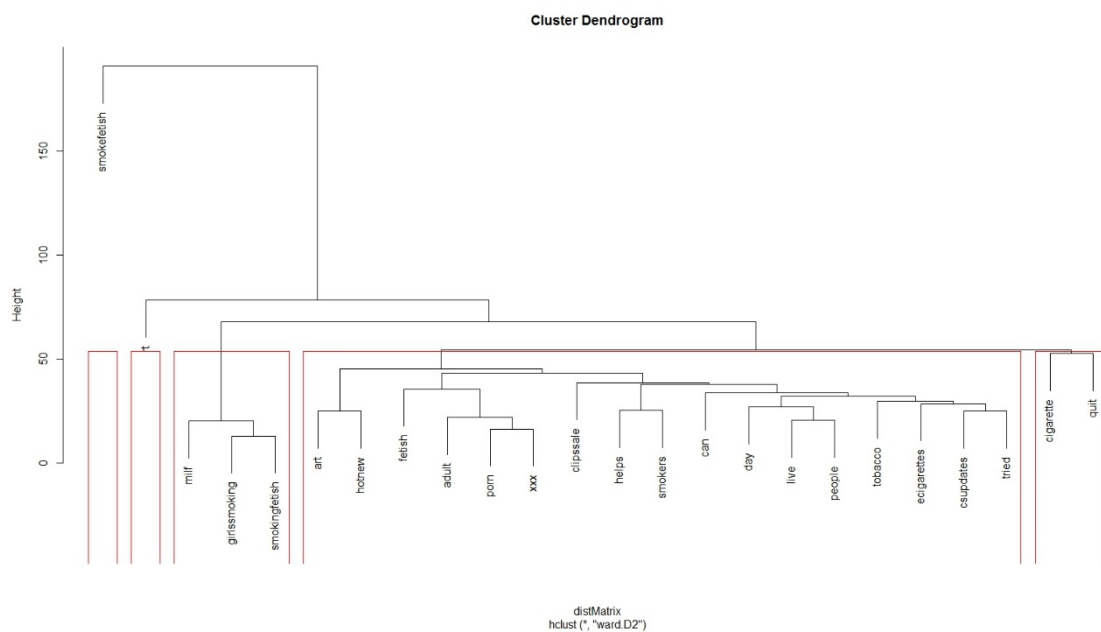
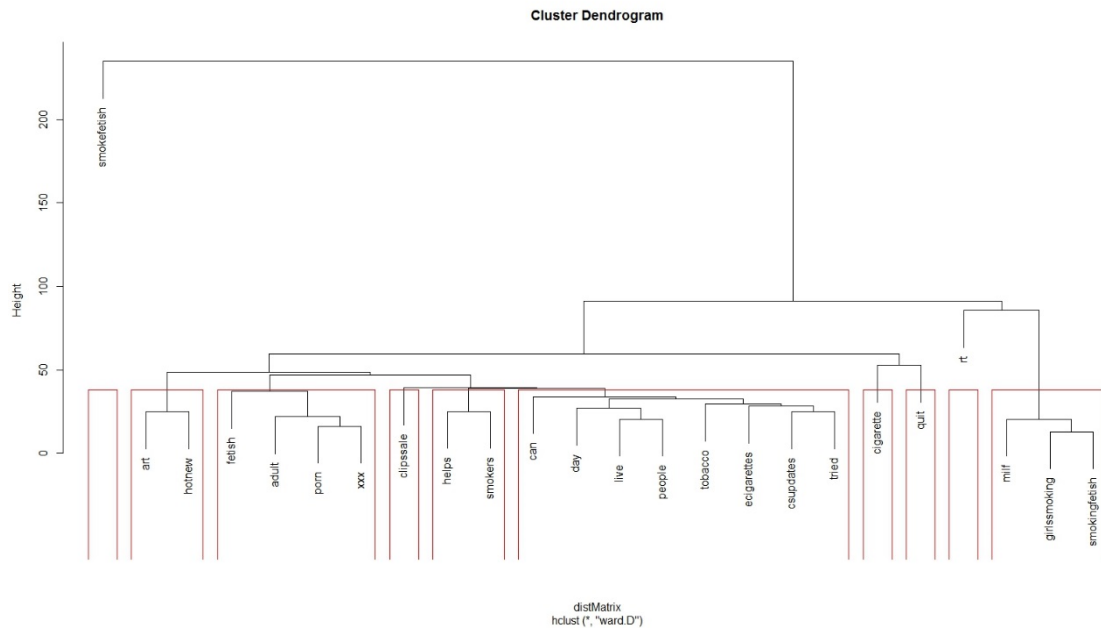
and  $w_{i \cup i'} = w_i + w_{i'}$  (4)

Contrary to Ward1, the input dissimilarities are Euclidean distances (not squared). They are squared within equation (4):  $\delta^2(i, i') = \sum_j (x_{ij} - x_{i'j})^2$ . It is such squared Euclidean distances that interest us, since our motivation arises from the error sum of squares criterion.

If members != NULL, then d is taken to be a dissimilarity matrix between clusters instead of dissimilarities between singletons and members gives the number of observations per cluster. This way the hierarchical cluster algorithm can be 'started in the middle of the dendrogram', e.g., in order to reconstruct the part of the tree above a cut. Dissimilarities between clusters can be efficiently computed only for a limited number of distance/linkage combinations, the simplest one being *squared* Euclidean distance and centroid linkage. In this case the dissimilarities between the clusters are the squared Euclidean distances between cluster means.

In hierarchical cluster displays, a decision is needed at each merge to specify which subtree should go on the left and which on the right. Since, for n observations there are n-1 merges, there are  $2^{n-1}$  possible orderings for the leaves in a cluster tree, or dendrogram. After ordering the subtree so that the tighter cluster is on the left (the last, i.e., most recent, merge of the left subtree is at a lower value than the last merge of the right subtree). Single observations are the tightest clusters possible, and merges involving two observations place them in order by their observation sequence number.

We use R to implement these methods, two of results are as following, the first using Murtagh (1985) and the second using Legendre (2012):



## 2.1.2 K-means clustering with R

**K-means** clustering is the most popular partitioning method. It requires the analyst to specify the number of clusters to extract. A plot of the within groups sum of squares by number of clusters extracted can help determine the appropriate number of clusters. Conceptually, the K-means algorithm:

Selects K centroids (K rows chosen at random)

Assigns each data point to its closest centroid

Recalculates the centroids as the average of all data points in a cluster (i.e., the centroids are p-length mean vectors, where p is the number of variables)

1. Assigns data points to their closest centroids
2. Continues steps 3 and 4 until the observations are not reassigned or the maximum number of iterations (R uses 10 as a default) is reached.
3. Implementation details for this approach can vary.
4. R uses an efficient algorithm by Hartigan and Wong (1979) that partitions the observations into k groups such that the sum of squares of the observations to their assigned cluster centers is a minimum. This means that in steps 2 and 4, each observation is assigned to the cluster with the smallest value of:

$$SS(k) = \sum_{i=1}^n \sum_{j=1}^p (x_{ij} - \bar{x}_{kj})^2$$

5. Where k is the cluster,  $x_{ij}$  is the value of the jth variable for the ith observation, and  $\bar{x}_{kj}$  is the mean of the jth variable for the kth cluster.
6. K-means clustering can handle larger datasets than hierarchical cluster approaches. Additionally, observations are not permanently committed to a cluster. They are moved when doing so improves the overall solution. However, the use of means implies that all variables must be continuous and the approach can be severely affected by outliers. They also perform poorly in the presence of non-convex (e.g., U-shaped) clusters.
7. The format of the K-means function in R is `kmeans(x, centers)` where x is a numeric dataset (matrix or data frame) and centers is the number of clusters to extract. The function returns the cluster memberships, centroids, sums of squares (within, between, total), and cluster sizes.

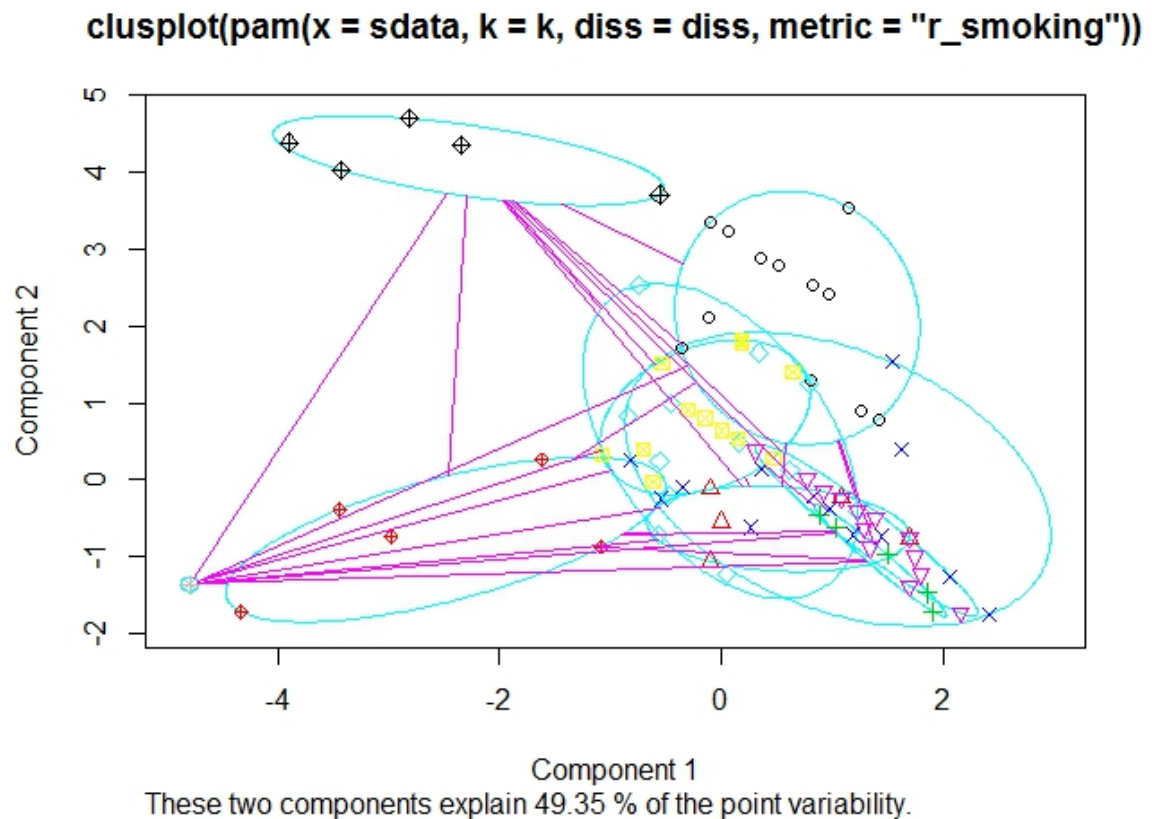
Since K-means cluster analysis starts with k randomly chosen centroids, a different solution can be obtained each time the function is invoked. Use

the `set.seed()` function to guarantee that the results are reproducible. Additionally, this clustering approach can be sensitive to the initial selection of centroids.

The `kmeans()` function has an `nstart` option that attempts multiple initial configurations and reports on the best one. For example, adding `nstart=25` will generate 25 initial configurations. This approach is often recommended.



And one sample result is as following:



### 3. Data Analysis Part

The goal of analysis of tweets is to cluster them and accurately label each one. The analysis is based on two parts.

First, we do Natural Language analysis on each tweets in two different properties. We consider that a tweet can be identify on the words which are most informative. We implement this in Term Frequency & Inverse Document Frequency (TFIDF). Here are some mathematical definition of TFIDF.

#### Term Frequency:

In the case of the term frequency  $tf(t,d)$ , the simplest choice is to use the *raw frequency* of a term in a document, i.e. the number of times that term  $t$  occurs in document  $d$ . If we denote the raw frequency of  $t$  by  $f_{t,d}$ , then the simple  $tf$  scheme is  $tf(t,d) = f_{t,d}$ . Other possibilities include

- Boolean "frequencies":  $tf(t,d) = 1$  if  $t$  occurs in  $d$  and 0 otherwise;

- Logarithmically scaled frequency:  $\text{tf}(t, d) = 1 + \log f_{t,d}$ , or zero if  $f_{t,d}$  is zero;
- augmented frequency, to prevent a bias towards longer documents, e.g. raw frequency divided by the maximum raw frequency of any term in the document:

$$\text{tf}(t, d) = 0.5 + \frac{0.5 \times f_{t,d}}{\max\{f_{t,d} : t \in d\}}$$

### Inverse Document Frequency:

The inverse document frequency is a measure of how much information the word provides, that is, whether the term is common or rare across all documents. It is the logarithmically scaled fraction of the documents that contain the word, obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient.

$$\text{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

With

- $N$ : total number of documents in the corpus  $N = |D|$
- $|\{d \in D : t \in d\}|$ : Number of documents where the term  $t$  appears (i.e.  $\text{tf}(t, d) \neq 0$ ). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to  $1 + |\{d \in D : t \in d\}|$ .

(From Wikipedia: <https://en.wikipedia.org/wiki/Tf%E2%80%93idf>)

In order to adapt this method with the clustering, we apply TFIDF along with cosine-similarity calculation.

The cosine of two vectors can be derived by using the Euclidean dot product formula:

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos \theta$$

Given two vectors of attributes,  $A$  and  $B$ , the cosine similarity,  $\cos(\theta)$ , is represented using a dot product and magnitude as

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

Source code is shown in python:

```
def __clean__(self, text):
    """
    Clean up a document through stemming and stop word removal.
    Stemming is the act of removing suffixes from a word to limit variation between verb tenses.
    Stop word remove is the act of removing common words from the document that likely play no meaning in the
    significance of the document.
    :param document: The document to be cleaned
    :return: The given document after stemming and stop word removal
    """
    text = re.sub("((http|https|ftp|ftps):\/\/[^\s_-+!*'(),$%&~.,:;]+)", '', text)
    text = re.sub("(#\[\s_\]]+)", '', text)
    text = re.sub("(#\[\s_\]]+)", '', text)
    text = re.sub("\p{P}+", '', text)
    text = re.sub("[\s]": "#", text)
    stopwords = ["a", "a's", "able", "about", "above", "according", "accordingly", "across", "actually", "after", "afterwards", "again", "against", "ain't",
    sb = []
    text = text.lower()
    re.sub(r'[\s_]+', '', text)
    for term in text.split():
        term = stem(term)
        if term not in stopwords:
            sb.append(term)
    return ' '.join(sb)
```

As can be seen from the code, we test a tweet after retrieving the most identical terms. Cleaned text will be used to calculate the TF and IDF for each term left.

```
def similar(self, label, document):
    # Calculate the Term Frequency - Inverse Document Frequency of each term in the corpus
    tfidf_by_term_by_document = {}
    tfidf_by_term = {}
    #tfidf_by_document = 0
    for term in document.text.split():
        tf = self.__tf__(term, document)
        idf = self.__idf__(term)
        tfidf_by_term[term] = tf * idf
        #tfidf_by_document += tfidf_by_term[term]
    tfidf_by_term_by_document[document] = OrderedDict(sorted(tfidf_by_term.items()))
    #return tfidf_by_document
```

The TFIDF evaluation will now formulate as a vector to test the cosine similarity between tweets to be tested and tweets in certain training set containing group of labeled tweets. If a tweet contains the identical terms belonging to certain group, it will show trend of larger similarity evaluation to that group.

```

corpus = []
total_sim = 0
for neighbor in self.documents:
    #define the total corpus of whole documents
    if neighbor.id == label:
        corpus.extend(neighbor.text.split())
for neighbor in self.documents:
    tfidf_by_term_neighbor = {}
    for term in neighbor.text.split():
        tf = self.__tf__(term, neighbor)
        idf = self.__idf__(term)
        tfidf_by_term_neighbor[term] = tf * idf
    tfidf_by_term_by_document[neighbor] = OrderedDict(sorted(tfidf_by_term_neighbor.items()))

    if neighbor.id == label:
        # Find the common terms in both the target and corpus
        common_terms = set(document.text.split()).intersection(set(corpus))

        tvector = [tfidf for term, tfidf in tfidf_by_term_by_document[document].iteritems() if term in common_terms]
        nvector = [tfidf for term, tfidf in tfidf_by_term_by_document[neighbor].iteritems() if term in common_terms]

        # Assert neither vector is empty
        if tvector and nvector:
            similarity = self.__cosine_distance__(tvector, nvector)
            if similarity > 0:
                total_sim += similarity
return total_sim

```

Second, we will move on to the sentimental analysis of feelings of twitterers when posting tweet. Since our categories are based on extent of positive, motive and negative, the sentiment of certain training set like positive smoking and quit smoking will definitely show differences. We implement the sentimental analysis by using Natural Language Toolkit based on Naïve Bayes Classifier. We need to extract features from the original text rather than cleaned text.

```

def extract_features(document):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['contains(%s)' % word] = (word in document_words)
    return features

```

Then we build up Naïve Bayes Classifier. Also we have definition here in Wikipedia:

### Naïve Bayes Classifier:

Abstractly, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector  $\mathbf{x} = (x_1, \dots, x_n)$  representing some  $n$  features (independent variables), it assigns to this instance probabilities

$$p(C_k | x_1, \dots, x_n)$$

For each of  $K$  possible outcomes or *classes*.

The problem with the above formulation is that if the number of features  $n$  is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' Theorem, the conditional probability can be decomposed as

$$p(C_k|\mathbf{x}) = \frac{p(C_k) p(\mathbf{x}|C_k)}{p(\mathbf{x})}.$$

We apply the mathematical model in python:

```
pos_tweets = []
neg_tweets = []
f = open('sentiment_training.txt', 'r')
line = f.readline()
while line != '':
    pos_pattern = re.compile('1\t')
    while pos_pattern.match(line):
        line = pos_pattern.sub('', line)
        pos_tweets.append([line, 'positive'])
        line = f.readline()
    neg_pattern = re.compile('0\t')
    while neg_pattern.match(line):
        line = neg_pattern.sub('', line)
        neg_tweets.append([line, 'negative'])
        line = f.readline()
    line = f.readline()
f.close()

# extract the word features out from the training data
word_features = get_word_features(get_words_in_tweets(tweets))

# get the training set and train the Naive Bayes Classifier
training_set = nltk.classify.apply_features(extract_features, tweets)
classifier = NaiveBayesClassifier.train(training_set)

def classify_tweet(tweet):
    return classifier.prob_classify(extract_features(tweet.split()))
```

Then we get the evaluation of the positive sense of a tweet. We import the probability as another dimension for tweets in case we can categorize tweets in the way we want.

Finally we apply the properties which have already been abstracted on numbers to k-Nearest Neighbor (kNN) Algorithm for classification of new tweet input.

Description of kNN from Wikipedia:

### K-Nearest Neighbor:

The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the classification phase,  $k$  is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the  $k$  training samples nearest to that query point.

A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the overlap metric (or Hamming distance).

As we have been able to build up mathematical model, we use Euclidean distance for kNN. Implementations are done in Python:

```
def classify0(inX, dataSet, labels, k):
    dataSetSize = dataSet.shape[0]
    diffMat = tile(inX, (dataSetSize, 1)) - dataSet
    sqDiffMat = diffMat**2
    sqDistances = sqDiffMat.sum(axis=1)
    distances = sqDistances**0.5
    sortedDistIndicies = distances.argsort()
    classCount={}
    for i in range(k):
        voteIlabel = labels[sortedDistIndicies[i]]
        classCount[voteIlabel] = classCount.get(voteIlabel,0) + 1
    sortedClassCount = sorted(classCount.iteritems(),
    key=operator.itemgetter(1), reverse=True)
    return sortedClassCount[0][0]

count = 0
count_exer = 0
for document in doc:
    count += 1
    count_inner = 0
    for category in categories:
        if count_inner == 0:
            sim = np.array([TrainSet.similar(category, document)])
        else:
            sim = np.append(sim, [TrainSet.similar(category, document)])
        count_inner += 1
    text = document.original_text
    text = re.sub("((http|https|ftp|ftps):\/\/[\\w$-_.+!*'(),$=]+)", '', text)
    text = re.sub("(@[\\w_]+)", '', text)
    text = re.sub("(#[\\w!$-_.+!*'(),$=]+)", '', text)
    text = re.sub("\\p{P}+", '', text)
    text = re.sub("[\\'\"':#,&!$]+", '', text)
    pos = classify_tweet(text).prob('positive')
    for category in categories:
        sim = np.append(sim, [pos])
    if count == 1:
        doc_sim = np.array([sim])
    else:
        doc_sim = np.append(doc_sim, [sim], axis = 0)























































    if kNN.classify0(sim, group, labels, 3) == 'pos_exercise':
        count_exer += 1
        if count_exer == 1:
            doc_exer = np.array([sim])
        else:
            doc_exer = np.append(doc_exer, [sim], axis = 0)
    print document.original_text
```

It will show result of classify for tweets of positive exercise. For now we have not integrate the result of analysis model with database or user interface. We are facing problem of curse of dimensionality. We are still optimizing the model and improving accuracy of analysis.

#### 4. Database Design

Then we are transmitting the tweets in MongoDB to MySQL. So far we have collected around 50000 tweets. Tweets relevant to our hashtags in consideration are stored in MongoDB initially. These tweets are later extracted into MySQL since it's easy to data analysis if the data is stored in a relational database. Also, It's easy to extract information unto User Interface if we have our data stored in MySQL. The UserID, TweetID, TweetText and CreationTime are stored in these tables.

Here is the snapshot of our database:

|                          |   |   | TweetID   | TweetText          | UserID  | CreateTime                     |
|--------------------------|---|---|---|--------------------|---|--------------------------------|
| <input type="checkbox"/> |    |    |    | 527566517672890368 | Pilateando en Olimpic II Ya no puede entrar nadie ... | 2358627272 2015-10-30 12:43:24 |
| <input type="checkbox"/> |    |    |    | 527566523301642240 | Pilateando en Olimpic II Ya no puede entrar nadie ... | 2358627272 2015-10-30 12:43:24 |
| <input type="checkbox"/> |    |    |    | 527566533946793984 | Buenas noches familiar #SoyDeAgua #shark #surf #wa... | 2358627272 2015-10-30 12:43:24 |
| <input type="checkbox"/> |    |    |    | 527566538640203776 | If you want to lose weight contact http://t.co/oRh... | 2365562754 2015-10-30 12:43:24 |
| <input type="checkbox"/> |    |    |    | 527566540531838977 | Because what's a day with out my weight loss coffe... | 88336290 2015-10-30 12:43:24   |
| <input type="checkbox"/> |    |    |    | 527566554267807744 | Entrenando con demasiado calor #GYM espalda y tric... | 2756704688 2015-10-30 12:43:24 |
| <input type="checkbox"/> |    |    |    | 527566558722146304 | #WeightLoss #Program Health Tip: Practice a Well-B... | 534928178 2015-10-30 12:43:24  |
| <input type="checkbox"/> |    |    |    | 527566559783313409 | #Healthy #Nutrition How to Lose Weight With Thermo... | 598370918 2015-10-30 12:43:24  |
| <input type="checkbox"/> |    |    |    | 527566559921729536 | #EmmaWatson #Fitness #Diet ###Fitness Fitness on t... | 559369958 2015-10-30 12:43:24  |
| <input type="checkbox"/> |    |    |    | 527566561129672704 | #FatLoss #Healthy 5 Secrets To Weight Loss While G... | 601794252 2015-10-30 12:43:24  |
| <input type="checkbox"/> |  |  |  | 527566562564132864 | #Workout #GetFit Tips to Lose Fat http://t.co/erCn... | 769756746 2015-10-30 12:43:24  |
| <input type="checkbox"/> |  |  |  | 527566563617284096 | Después de una sesión de gym toca una duchita fres... | 573865263 2015-10-30 12:43:24  |
| <input type="checkbox"/> |  |  |  | 527566565911179264 | #JohnMayer #Fitness Paleo Diet Success Stories htt... | 784182600 2015-10-30 12:43:24  |
| <input type="checkbox"/> |  |  |  | 527566584429428736 | 8 Min Abs The Classic - Level 1 #workout #abs #six... | 189855189 2015-10-30 12:43:24  |
| <input type="checkbox"/> |  |  |  | 527566588455559168 | Bem melhor assim... #Run #Mar #Floripa #VemVerão #... | 98040902 2015-10-30 12:43:24   |
| <input type="checkbox"/> |  |  |  | 527566595267510274 | Hoy recorrimos 10 kilómetros de #running con temas... | 384937373 2015-10-30 12:43:24  |
| <input type="checkbox"/> |  |  |  | 527566634643632129 | Before And After Weight Loss Pictures Women   http... | 826488764 2015-10-30 12:43:24  |
| <input type="checkbox"/> |  |  |  | 527566635734159360 | RT @MisTillas: Nike LunarGlide 5 de ricardo_spv #M... | 83247301 2015-10-30 12:43:24   |

#### 5. User Interface Section

Here are parts of our websites showing some of our features.



Figure 1. Homepage (1)

When users enter our websites, they will see a welcome interface.

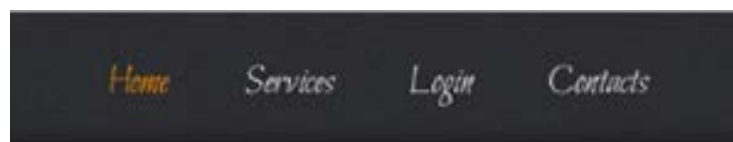


Figure 2. Navigation

Users can choose to login by click "Login" on the top right of the web to find more personalized detail. And they can find out main features of our project after clicking "Services". And "Home" will allow users to return the home page.





Presented by Rutgers students.  
Software Engineering Group#3

Follow Us:   

Figure 3. Homepage (2)

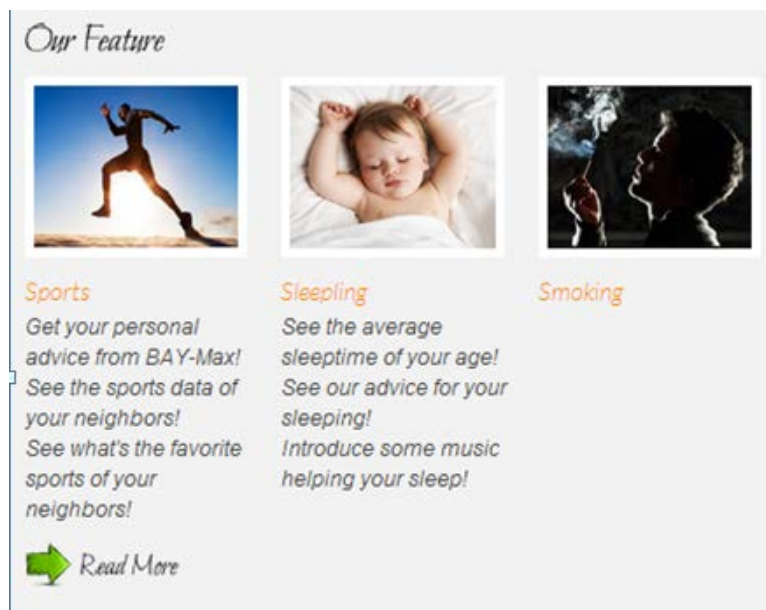


Figure 4. Choose feature

Users are also available to access those features from links below the welcome interface.



Figure 5. Calendar

Users will find out recent events happened nearby from the calendar displayed above, and figure out what they are interested in then add it to their own schedule.

Here are some code about calender:

```
<div class="grid_4">
  <h2 class="top-1">Latest Events</h2>
  <div class="box-4 top-2">
    <div class="date"><strong>29</strong><span>May, 2015</span></div>
    <p><a href="#" class="link">Lorem ipsum dolor sit amet consetetur</a><br>Sadipscing elitriam nonumy
  </div>
  <div class="box-4 top-3">
    <div class="date"><strong>21</strong><span>May, 2015</span></div>
    <p><a href="#" class="link">Dolore magna aliquyam erat dolor</a><br>At vero eos et accusam et justo
  </div>
  <div class="box-4 top-4">
    <div class="date"><strong>21</strong><span>May, 2015</span></div>
    <p><a href="#" class="link">Dolore magna aliquyam erat dolor</a><br>At vero eos et accusam et justo
  </div>
  <a href="#" class="button top-4">Read More</a>
</div>
<div class="clear"></div>
</div>
</section>
```



Figure 6. Services page

Users can click “Services” and enter this page.



Figure 7. Heat map

On this page, users are able to find a heat map showing the density of, for example, people who exercise or smoke of different regions.

Start-Up

[Home](#)
[Services](#)
[Login](#)
[Contact](#)

How to Find Us

Science and Engineering Resourc...

View larger map

*If you have any question or you want get more professional advice, Please feel free to contact us!*

**Tips: Know more about sub-health.**  
Suboptimal health can be defined as a state characterized by some disturbances in psychological behaviors or physical characteristics, or in some indices of medical examination, with no typical pathologic features, which exactly characterize common situation that most of people are facing.

Contact Form

Name

Email

Phone

Message

Clear

Send

Presented by Rutgers students.

Software Engineering Group#3

Follow Us:

Figure 8. Contact us page

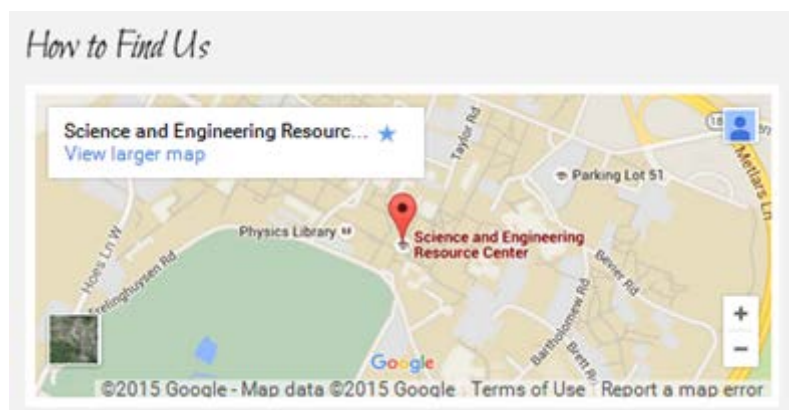


Figure 9. Map

Users will enter this page after clicking “Contacts” and they can easily find the location of our team from the map.

*Contact Form*

Name

Email

Phone

Message

Clear Send

Figure 10. Contact Form

And they can also write down advises about our project on the contact form showed at the right of this website.

\*\*Here are some codes about heatmap;



Figure 7. Heat map

On this page, users are able to find a heat map showing the density of, for example, people who exercise or smoke of different regions.

The codes are as follows:

```

<script src="https://maps.googleapis.com/maps/api/js?v=3.exp&libraries=visualization"></script>
<script type = "text/javascript" src = "jsonGeoData(Sample).js"></script>
<script type = "text/javascript" src = "PerCapitaHeatMap.js"></script>

```

```

function PushGeoData(geoData)
{
    for(var i = 0; i < geoData.length; i++)
    {
        var latLng = new google.maps.LatLng(geoData[i].lat, geoData[i].log);
        var weightedLoc =
        {
            location: latLng,
            weight: geoData[i].weight
        };
        var regularLoc =
        {
            location: latLng,
            weight: 1
        };

        WeightedMapData.push(weightedLoc);
        RegularMapData.push(regularLoc);
    }
}

```

```

function initializeHeatMap()
{
    var mapOptions = {
        center: new google.maps.LatLng(39.833333, -98.583333),
        zoom: 4,
        mapTypeId: google.maps.MapTypeId.SATELLITE
    };

    var mapCanvas = document.getElementById('map_canvas');

    map = new google.maps.Map(mapCanvas, mapOptions);

    heatmap = new google.maps.visualization.HeatmapLayer({
        data: HeatMapData,
        radius: 30
    });
    heatmap.setMap(map);
}

google.maps.event.addDomListener(window, 'load', initializeHeatMap);
PushGeoData(JsonGeoData);

```

And the data sample is as follows:

```
1 jsonGeoData = [{lat:-6.106558,log:105.881426,weight:8.36371},
2 {lat:14.494622,log:121.056881,weight:8.36371},
3 {lat:55.614453,log:-4.546886,weight:8.36371},
4 {lat:33.426617,log:-84.016211,weight:262.542997321047},
5 {lat:-23.572625,log:-46.696551,weight:8.36371},
6 {lat:33.81255,log:-117.835573,weight:55.0813764170109},
7 {lat:51.553442,log:0.631328,weight:8.36371},
8 {lat:38.828829,log:-76.996461,weight:23.6348904762913},
9 {lat:51.872421,log:-8.515075,weight:8.36371},
10 {lat:-2.72827,log:107.62923,weight:8.36371},
11 {lat:-6.23713,log:106.74764,weight:8.36371},
12 {lat:-6.13165,log:106.77939,weight:8.36371},
13 {lat:-7.755136,log:110.381711,weight:8.36371},
14 {lat:-6.26458,log:106.83736,weight:8.36371},
15 {lat:14.573379,log:121.059279,weight:8.36371},
16 {lat:23.518996,log:-75.75598,weight:8.36371},
17 {lat:-6.21973,log:106.83159,weight:8.36371},
18 {lat:-6.28172,log:106.65331,weight:8.36371},
19 {lat:33.498813,log:-84.366633,weight:147.342977342407},
20 {lat:33.498781,log:-84.366584,weight:147.336710843836},
21 {lat:48.871092,log:2.304153,weight:8.36371},
22 {lat:38.794405,log:-112.039254,weight:210.798287600421},
23 {lat:-6.347213,log:106.673026,weight:8.36371},
24 {lat:20.754341,log:-103.384389,weight:8.36371},
25 {lat:18.71362,log:98.976572,weight:8.36371},
26 {lat:6.996687,log:100.476818,weight:8.36371},
27 {lat:40.590218,log:-74.146715,weight:4.25985887156895},
28 {lat:32.592948,log:-92.070496,weight:627.793705779366},
29 {lat:41.582899,log:-72.95847,weight:58.5999027041238},
30 {lat:36.158973,log:-115.331681,weight:22.5686859465963},
31 {lat:33.839609,log:-83.543418,weight:134.75454624861},
32 {lat:33.576728,log:-112.180412,weight:18.4935919694346},
33 {lat:-1.449888,log:-48.492938,weight:8.36371},
34 {lat:29.476211,log:-95.054756,weight:159.404075260761},
35 {lat:33.811575,log:-118.389283,weight:43.1646921580777},
36 {lat:35.354283,log:-119.092578,weight:61.4365588186222},
37 {lat:38.876147,log:-99.330331,weight:410.616311628098},
38 {lat:40.327662,log:-76.20992,weight:43.1955462391277},
```