# Big Data Weather Analysis

Felipe de Morais and Michael Sobelman
https://github.com/mbs9b7/BigData

## 1. Introduction

Searching for some ideas for our project, we found a paper describing a huge dataset about global weather. We started to look deeper on this dataset, and we discovered that this dataset has a historical data covering terrestrial air temperature [1] and precipitation [2] from 1900 to 2010. The dataset is split in two datasets, one just for air temperature and the other just for precipitation. The format for each dataset is the same. It contains one file with information about each year. Each file contains a bunch of lines with the latitude and longitude of some place around the world, the average temperature or average precipitation per month. This dataset also has global information. However, for our project we are going to use just the terrestrial one. We had some questions about the data, and we started wondering if we could answer that questions and how could we answer them. We first look to the air temperature dataset and we realized that it had a huge amount of data. Thus, we decided to start by solving smallest problems.

The air temperature dataset is divided on a set of files. Each file contains the data for each year from 1900 to 2010. Also, each file contains the information of almost 1 million locations around the world. The whole dataset has more than 1Gb of data. After realizing all these issues, we really thought that this is a big data problem. We started our searches and tests on a single file, the most updated file of the set, that is the 2010. When we started to analyse this file, we realized that the format of it was very strange. In order to read it properly, we had to discover how the file was formatted and then fix it. The problem was that the columns were separated by a random number of spaces. Also, the negative sign in front of some values make the first column to have some extra space at the very beginning of the line. After identifying all these format errors, we start to search how to fix it.

The initial idea was to ask some questions and answer them by searching on the dataset. However, we realized that the dataset was really huge to run a query at every time. It would take too much time. Thus, we came up with the idea of loading all the data into Hadoop and execute some predefined queries on it. After having the results of these queries, we could store it on a relational database in order to retrieve the data from a web page, and it would be very fast. The tools we have used to achieve this idea were HDFS, Hadoop, Hive, Sqoop, MySQL, PHP, and the Google API. We use Hadoop and HDFS to store the data while Hive runs queries on it. Those results are then transferred to MySQL using Sqoop. From there we use PHP to access the results from MySQL and then display them on a map using the Google Maps API.

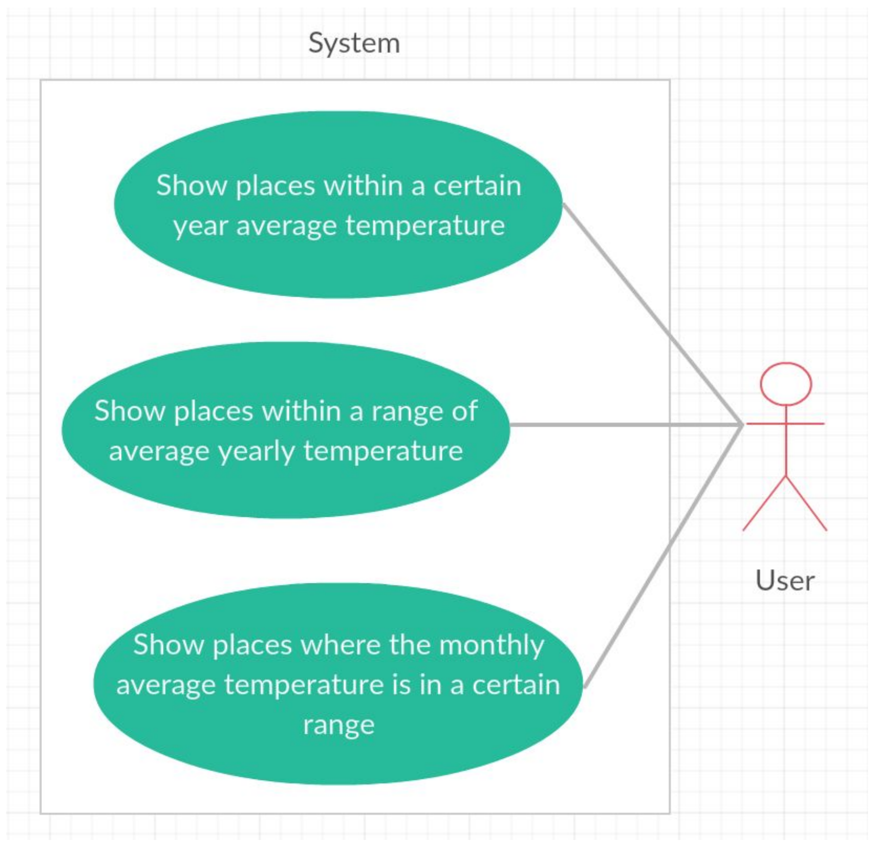
## 2. Objectives

The goal of this project is to present the weather data to the user in a way that is both readable and accessible.To do that, we settled on making a web app which would display weather data on a map. Thus, we can is to analyse this data to answer some questions. For example: What could be a good place to live in the world if we consider the best average temperature equal to 25 degrees Celsius?, What are the places in the world where the yearly average temperature is between -40 to -30 degrees Celsius?, I would like to go on vacation in a place where I can predict the temperature. Is there any place in the world where the monthly average temperature is between 20 to 30 degrees Celsius?, and so on.

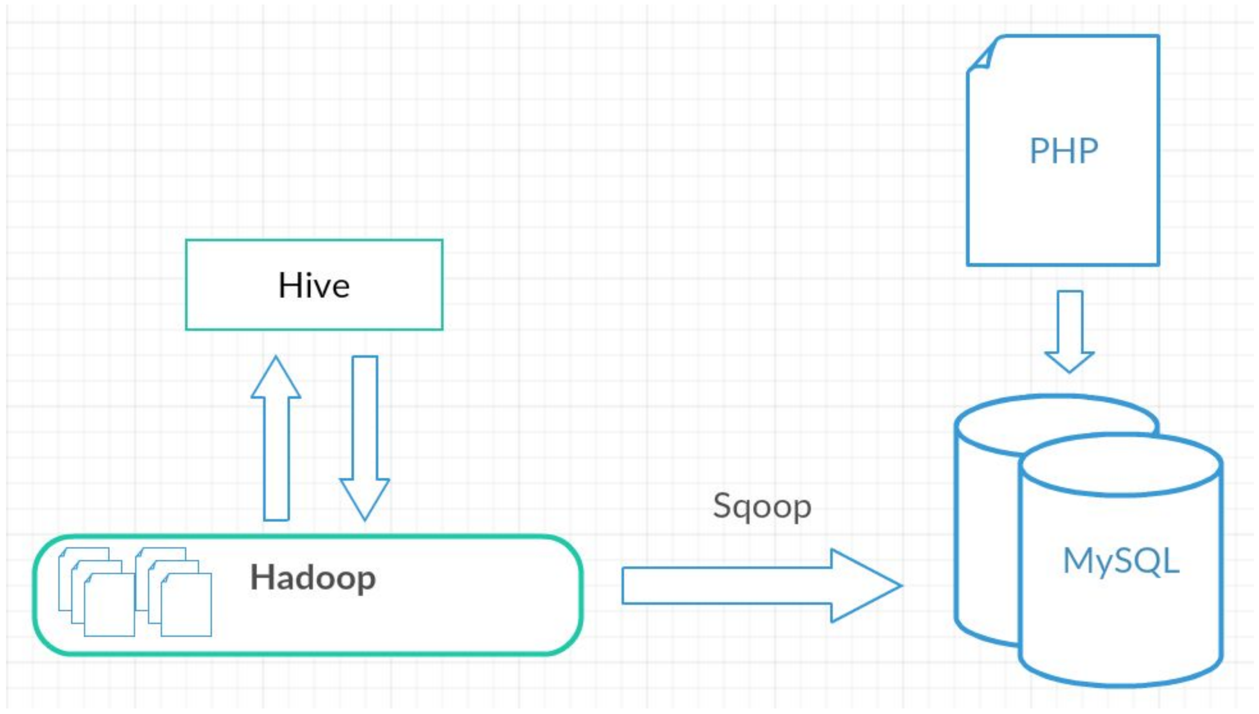
## 3. System Requirements

In order to run our queries we need a cluster running Hive through Hadoop. We then need MySQL to store the results of the query and Sqoop to move the results from HDFS to MySQL. We then use an Apache web server to display the data on a webpage. That webpage uses PHP to access the data in our MySQL database, and through the use of the Google Maps API (which is built with JavaScript) we display that data on a map.

## 4. Use Case Diagram


System
Show places within a certain year average temperature
Show places within a range of average yearly temperature
Show places where the monthly average temperature is in a certain range
User

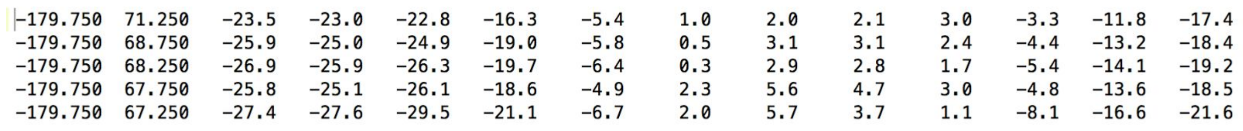
## 5. Design and Methodology



This diagram shows our design. We used Hive to make our queries. Hive then stores those queries in HDFS. We then used Sqoop to move those to MySQL and displayed them with PHP.

## 6. Data Structure

In order to simplify the problem, this project was based just on the Air Temperature dataset. This dataset has more than 1Gb of size. It is divided in files. Each file stores the data from a different year, that are from 1900 to 2010. Each file contains 14 columns. The first two columns store the longitude and latitude, respectively. The other 12 columns store the average air temperature for each month of the year. We can see the format of some lines of one file on the image below.



```
-179.750  71.250   -23.5   -23.0   -22.8   -16.3    -5.4     1.0     2.0     2.1     3.0    -3.3   -11.8   -17.4
-179.750  68.750   -25.9   -25.0   -24.9   -19.0    -5.8     0.5     3.1     3.1     2.4    -4.4   -13.2   -18.4
-179.750  68.250   -26.9   -25.9   -26.3   -19.7    -6.4     0.3     2.9     2.8     1.7    -5.4   -14.1   -19.2
-179.750  67.750   -25.8   -25.1   -26.1   -18.6    -4.9     2.3     5.6     4.7     3.0    -4.8   -13.6   -18.5
-179.750  67.250   -27.4   -27.6   -29.5   -21.1    -6.7     2.0     5.7     3.7     1.1    -8.1   -16.6   -21.6
```

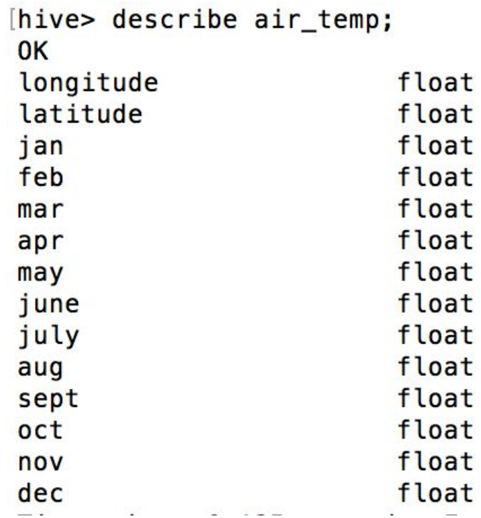
This image just show some lines of one file. Each file has almost a million of lines. Each line represents the monthly average temperature of a place in the world. These files uses spaces to separate the columns. The problem of it is that the number of spaces is not fixed. Before we start working with this data, we had to come up with a way to reformat the files. We use some regular expressions to replace the spaces by comma. Thus, the file was separated by commas, and it was ready to work with.

## 7. Implementation

As it was shown on the design section, our implementation uses Hive to load the data into Hadoop and to execute the queries. The first thing we did was to create a table on Hive. For that, we used the following command.

```
CREATE TABLE air_temp(longitude FLOAT, latitude FLOAT, jan FLOAT, feb FLOAT, mar
FLOAT, apr FLOAT, may FLOAT, june FLOAT, july FLOAT, aug FLOAT, sept FLOAT, oct
FLOAT, nov FLOAT, dec FLOAT) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

The resulting table on Hive is shown on the image bellow.



```
hive> describe air_temp;
OK
longitude               float
latitude                float
jan                     float
feb                     float
mar                     float
apr                     float
may                     float
june                    float
july                    float
aug                     float
sept                    float
oct                     float
nov                     float
dec                     float
```

After having the table created, we loaded the data into this table by using:

```
LOAD DATA LOCAL INPATH '/home/hadoop/air_temp/air_temp.2010' OVERWRITE INTO
TABLE air_temp;
```

In this command, we are using just one file for the air temperature of 2010. This command will load the data from the local storage and store it on Hadoop using Hive. After having the data on Hadoop, we have started to execute some queries. For example, this query select longitude and latitude of the first 10 rows on our data:

```
SELECT longitude, latitude FROM air_temp LIMIT 10;
```

However, just execute the queries was not enough for our problem. We had to come up with a way to export this result and store it. Thus, we start to look on Hive documentation and we discovered that we can store the result of our queries in files on Hadoop. In order to accomplish that, we used this command:

```
INSERT OVERWRITE DIRECTORY '/user/hadoop/output' SELECT printf(“%f,%f”, longitude,
latitude) FROM air_temp LIMIT 10;
```

This command is going to do exactly the same as the last command when we are talking about selecting data. However, this one is going to store the resulting data into Hadoop in a folder called output instead of showing the result on the console. Also, we are using a User-Defined Function (UDF) in order to format the resulting file. The function is called 'printf' and it receives two string inputs and format it to 'string1,string2', where string1 is the first argument, and string2 is the second argument. The resulting file will have this format for each line.

Once we have the queries working and storing the results, we started to answer our initial questions. The first question was: What could be a good place to live in the world if we consider the best average temperature equal to 25 degrees Celsius? The answer for this question was found by executing this query:

```
INSERT OVERWRITE DIRECTORY '/user/hadoop/output' SELECT printf(“%f,%f”, longitude,
latitude) FROM air_temp where (jan+feb+mar+apr+may+june+july+aug+sept+oct+nov+dec)/12
== 25;
```

This command select the longitude and latitude of the places where the average is equals to 25. It has to calculate the average by summating the temperature of all the months and dividing it by 12 (number of months in a year). As it was explained before, this command will also store this result in a formatted file on Hadoop.

The second question was: What are the places in the world where the yearly average temperature is between -40 to -30 degrees Celsius? The answer for this question came from this query:

```
INSERT OVERWRITE DIRECTORY '/user/hadoop/output' SELECT printf(“%f,%f”, longitude,
latitude) FROM air_temp where (jan+feb+mar+apr+may+june+july+aug+sept+oct+nov+dec)/12
>= -40 AND (jan+feb+mar+apr+may+june+july+aug+sept+oct+nov+dec)/12 <= -30;
```

On this command, we calculate the average twice in order to search for a range and not just a single value. Again, the result is going to be stored in Hadoop. Finally, the last question was: I would like to go on vacation in a place where I can predict the temperature. Is there any place in the world where the monthly average temperature is between 20 to 30 degrees Celsius? The answer was found using this query:

```
INSERT OVERWRITE DIRECTORY '/user/hadoop/output' SELECT printf(“%f,%f”, longitude,
latitude) FROM air_temp where jan >= 20 AND jan <= 30 AND feb >= 20 AND feb <= 30 AND
mar >= 20 AND mar <= 30 AND apr >= 20 AND apr <= 30 AND may >= 20 AND may <= 30
```

```
AND june >= 20 AND june <= 30 AND july >= 20 AND july <= 30 AND aug >= 20 AND aug <=
30 AND sept >= 20 AND sept <= 30 AND oct >=20 AND oct <= 30 AND nov >= 20 AND nov <=
30 AND dec >= 20 AND dec <= 30;
```

This command has to check the range of each month separately. This turned to be a huge query with so many AND operators. By having this data and the queries, we were able to answer our questions. Now, the idea is to show this answers in a Web app. In order to accomplish this task, we export the data from Hadoop to MySQL and by using PHP we are able to request this data very easily. To export the data from Hadoop to MySQL we have used Sqoop. Before we export the data, we had to create a table on MySQL to store this new data. One example of a table that we have create is the following:

First we create a database called weather:

```
create database weather;
```

Then, we select this database:

```
use weather;
```

Finally, we create a table to store the results for the our first question.

```
create table temp25 (longitude float(6,3),latitude float(6,3),PRIMARY KEY(longitude, latitude));
```

After having the table on MySQL to store the results of our first query, we were able to export the data from Hadoop to MySQL using Sqoop. The command used is the following:

```
sqoop export --connect jdbc:mysql://localhost/weather --username root —password password
--table temp25 --export-dir /user/hadoop/output
```

Once the data is on MySQL we can easily retrieve this data by using PHP. When a query is first run on our website, we use the Google Maps API to render a map on our site. Next, once the first set of longitude and latitude values are stored as variables in PHP, we again use the Google Maps API to put a pin in the map on that location. We then loop through the rest of the results, putting more pins down at each location. The result is a nice map with pins in all the locations that match the results of the query.

## 8. Testing and Validation

In order to check our results, we made some queries on our site, and then ran a select statement on the raw data. We then did the math to make sure that the query was returning the proper data.

## 9. Conclusions

One of the first things we noticed was the change in temperature over the years. Most areas have gotten warmer (due to global warming) so we came to the conclusion that using an average temp for the last 100 years was probably not an accurate representation of future data.

We also noticed that averages alone don't necessarily tell the whole story, as some places have a wide range of temperatures throughout the year. For this reason we added the "No Month Outside of Range" button which only looks at areas in which no month is outside of a specified range.

We also made a few conclusions regarding our software stack. At one point we put the data set for 1 year (more than 85,000 rows) into MySQL to test a few queries while we were having problems with Hive, and to our surprise, MySQL ran the query in well under 0.1 second, compared to Hive which took significantly longer. At this point we decided to switch to using a MySQL Database to directly run the queries because MySQL can easily query the data we are using. However, if we decide to use a dataset containing multiple years, it would be inefficient to execute it using just MySQL, so we would switch back to using Hive to run the query.

## 10. References

[1] Willmott, C. J. and K. Matsuura (2012) Terrestrial Precipitation: 1900-2010 Gridded Monthly Time Series.
http://climate.geog.udel.edu/~climate/html_pages/Global2011/Precip_revised_3.02/README.GlobalTsP2011.html

[2] Willmott, C. J. and K. Matsuura (2012) Terrestrial Temperature: 1900-2010 Gridded Monthly Time Series.
http://climate.geog.udel.edu/~climate/html_pages/Global2011/README.GlobalTsT2011.html

## 11. User Manual (and steps to run your program and system)

This manual describes all the steps we have done to run our demo on the final presentation. It is important to be clear that we have two versions of our application. The first one uses Hive to store and execute the queries, and sqoop to export the data to MySQL. The second one, we just load the data into MySQL and we execute the queries on MySQL. The main reason why we are using this directly on MySQL is that in order to run Hive, Hadoop, Sqoop, and to store the whole dataset, we were using a virtual machine on AWS, but we ran out of credits. We then tried to run it on our personal VMs, but we were not able to accomplish that due to a lack of resources on the local machine.

a) On a computer with GitHub installed, run this command to clone our repository onto your computer. You should probably do this in a directory that can be displayed via Apache (i.e. public_html)

```
git clone https://github.com/mbs9b7/BigData.git
```

b) Open MySQL and in a database of your choice (we used one called weather) create a table using this command

```
    CREATE TABLE air_temp (longititude float(6,3), latitude float(6,3), jan float(3,1), feb
float(3,1), march float(3,1), april float(3,1), may float(3,1), june float(3,1), july float(3,1), aug
float(3,1), sept float(3,1), oct float(3,1), nov float(3,1), decm float(3,1), PRIMARY
KEY(longititude, latitude));
```

c) Use this command to insert the air_temp.2010 file into MySQL

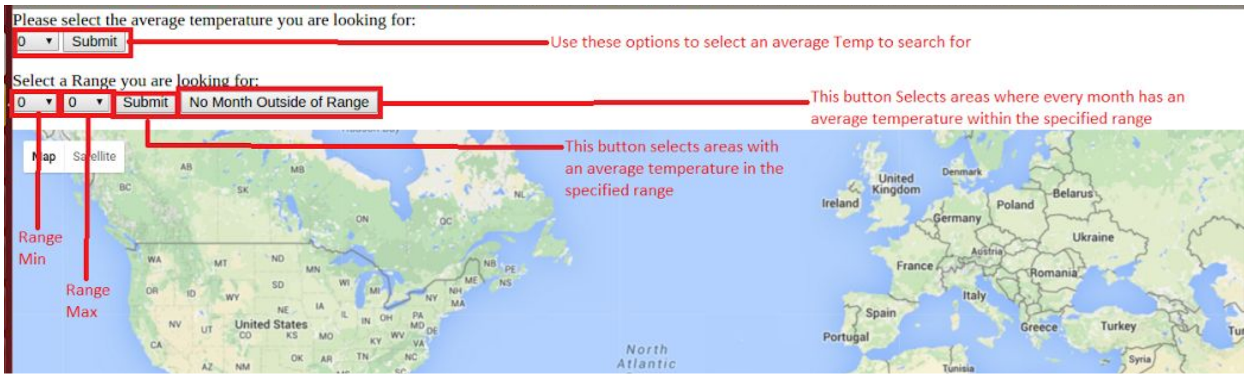
```
    LOAD DATA INFILE '<directory of your BigData Repo>/air_temp.2010' INTO TABLE
air_temp fields terminated by ',' lines terminated by '\n';
```

d) Edit your index.php file to correctly connect to your database
   i) Mine uses a file located at "../secure/database.php" which is set up to fill in the values for connecting to MySQL as such:

```
<?php
        DEFINE("HOST","localhost");
        DEFINE("DBNAME","weather");
        DEFINE("USERNAME","root");
        DEFINE("PASSWORD","password")
?>
```

   but you can just remove the include statement on line 169, and edit the $conn variable on line 170

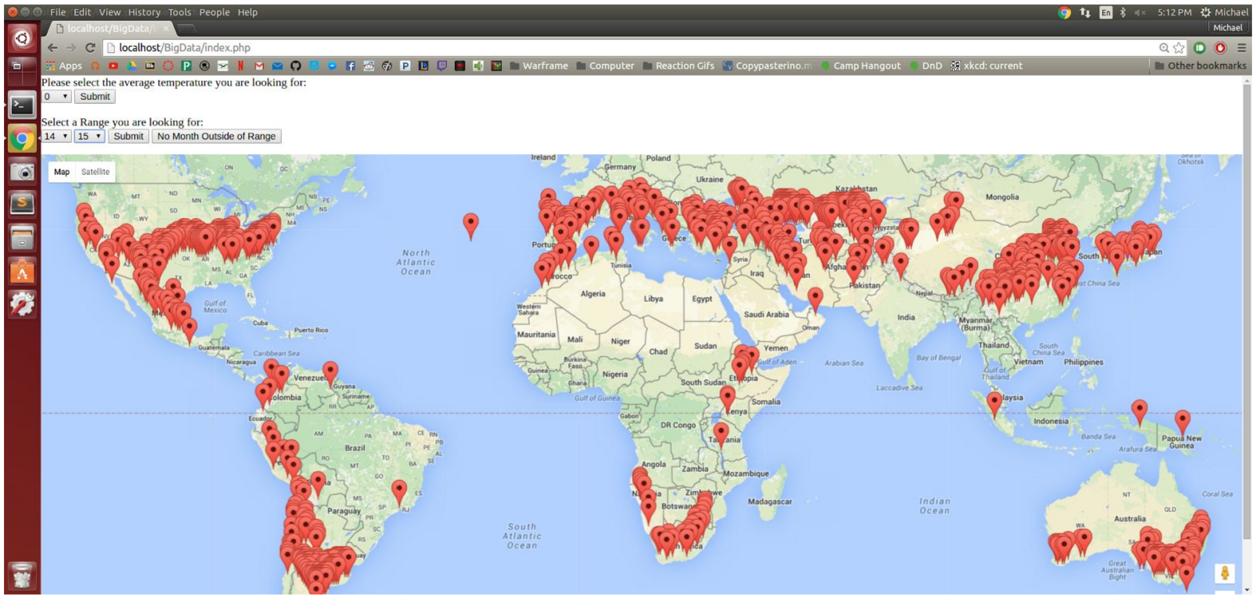
e) Access index.php from your browser of choice. (make sure php5 and Apache2 are installed and configured correctly) It should look like this:


Please select the average temperature you are looking for:

Select a Range you are looking for:

0
Submit
Use these options to select an average Temp to search for
0
0
Submit
No Month Outside of Range
This button Selects areas where every month has an average temperature within the specified range
This button selects areas with an average temperature in the specified range
Map
Satellite
Range Min
Range Max
United States
North Atlantic
Ireland
United Kingdom
Denmark
Poland
Belarus
Germany
Ukraine
France
Austria
Romania
Italy
Spain
Portugal
Greece
Turkey

When you search a query, all areas that meet the requirements will be plotted as pins on the map.


Please select the average temperature you are looking for:
0 Submit

Select a Range you are looking for:
14 15 Submit No Month Outside of Range

This Query shows areas with yearly average temperatures between 14℃ and 15℃.
All points represent an area of 0.250 degrees of latitude x 0.250 longitude.