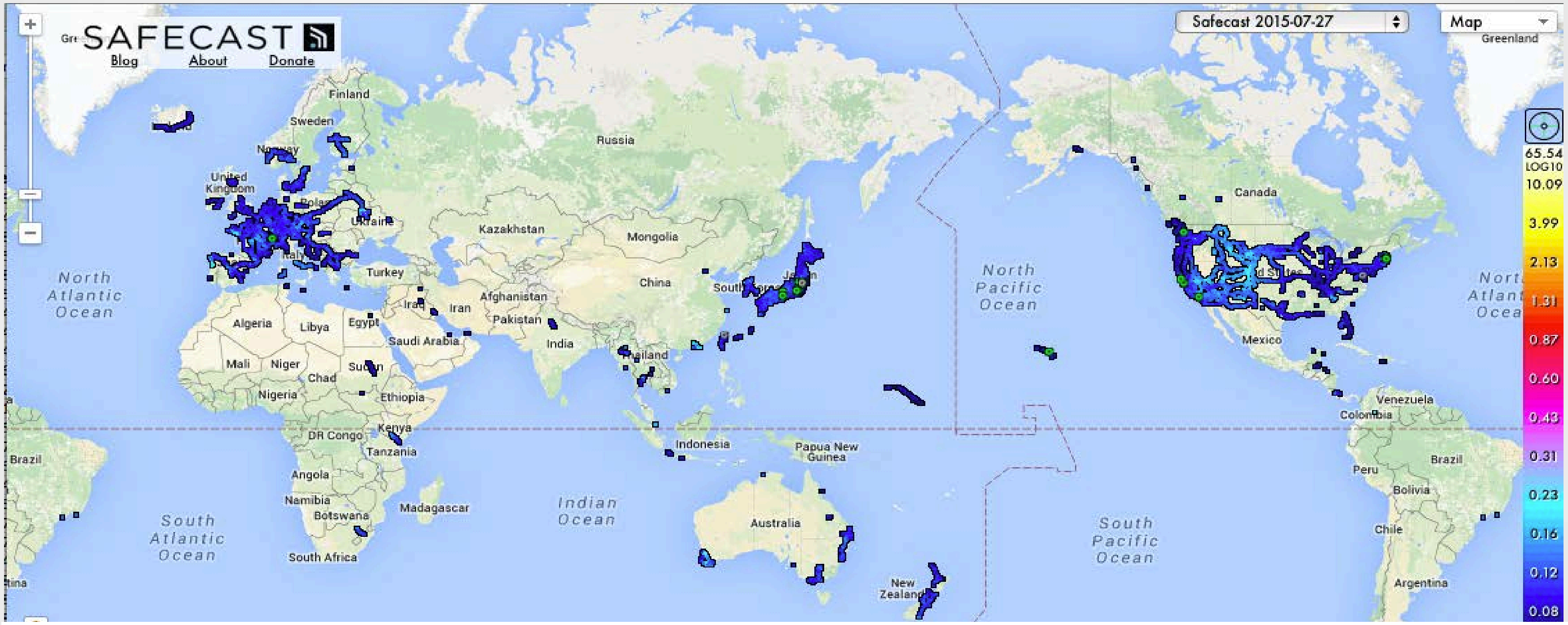
Advanced Geospatial Data Analytics: Pig and Python on Hadoop

Exploring radiation data into HDFS

- Before you begin:
  - The data for the lab is stored in /gpfs_scratch/geog479/lecture7
  - Copy the data to your home directory:
  - cp -r /gpfs_scratch/geog479/lecture7 ~/

Data 1: Safecast dataset



CapturedTime	Latitude	Longitude	Value	Unit	LocationName	DeviceID	MD5Sum	Height	Surface	Radiation	UploadedTime	LoaderID
6/14/13 23:20	37.36375667	140.3273567	137	cpm			47b0640bf4b1bdbb590576e7cce40c4a				27:47.6	
6/14/13 23:18	37.36884833	140.3483217	126	cpm			161afffdc10e0a31fdc995accdd8c507				27:47.6	
6/14/13 23:16	37.36668833	140.3486633	91	cpm			b6d9a1cdf8f41efc48bc9c4c5475e276				27:47.6	
6/14/13 23:15	37.36305667	140.3497933	112	cpm			c71d2f4ee1b5646a3dca1a4d80bebe3a				27:47.6	
6/14/13 23:16	37.36308833	140.349405	103	cpm			f63865c98d3fad21faaefd6780c70ef2				27:47.6	
6/14/13 23:16	37.36305333	140.3494183	114	cpm			233865c2221eb8671595beef8a9c0697				27:47.6	
6/14/13 23:15	37.36304333	140.3493883	113	cpm			32e9742206fb6928aee91cc401210574				27:47.6	
6/14/13 23:15	37.36304667	140.3493967	114	cpm			56679a4906c54a01863e47bb6a9b1c29				27:47.6	
6/14/13 23:16	37.36604667	140.3487767	95	cpm			a52e0252721492f8a263bea41832fee4				27:47.6	
6/14/13 23:16	37.36386167	140.3492317	91	cpm			a8898be32433cda2ad37c3b46630b787				27:47.6	
6/14/13 23:16	37.36322333	140.3493683	96	cpm			29c7b4011a320d8f335b553fc2a1ff36				27:47.6	
6/14/13 23:16	37.36351833	140.3493117	92	cpm			50dfca2c9db898de8ec2cc11f3682a13				27:47.6	
6/14/13 23:16	37.365425	140.3489267	86	cpm			6712f242f6ae678e2d89e1c701d27470				27:47.6	
6/14/13 23:16	37.36427833	140.3491667	91	cpm			4fd9184a51c997b5b176b1d64525de6b				27:47.6	

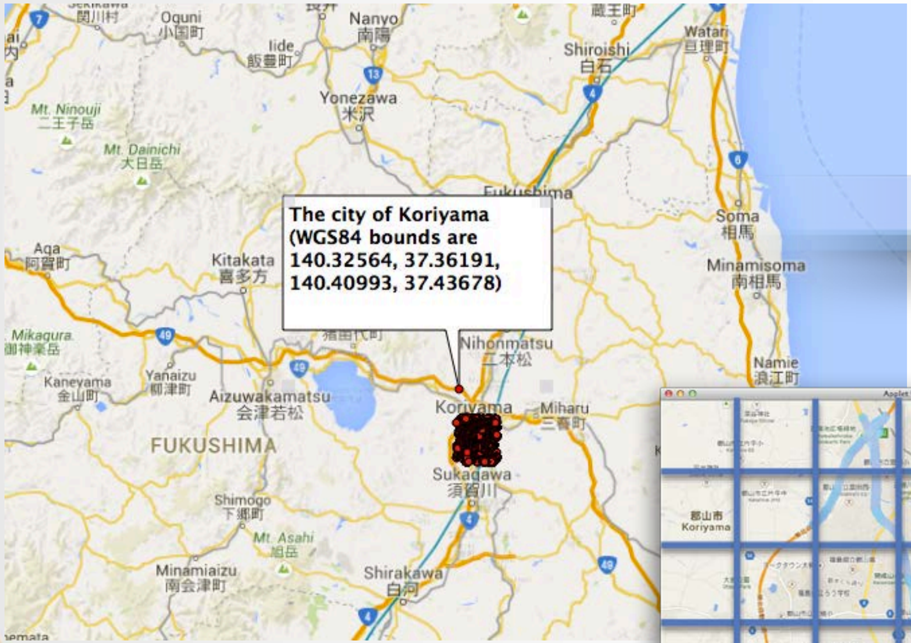
Safecast – http://blog.safecast.org

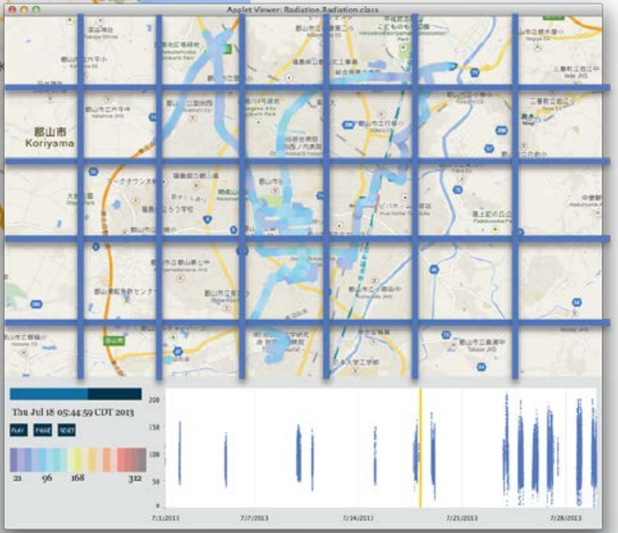
Case 1: Making a heatmap input data based on radiation dataset

- Objective:
  - Use Pig to navigate through HDFS and explore a dataset.
  - Make a heatmap data for generating a heatmap image file.

- Before you begin:
  - Radiation data is in ~/lecture7/data
  - Load the radiation data to HDFS
  - Optional: check the replication factor we had talked about
    - hdfs dfs -ls
    - hdfs dfs -setrep –w 2 radiation.csv
    - Now check the HDFS to see what happens

The city of Koriyama, Japan




Applet Viewer: Radiation.Radiation.class
Koriyama
Thu Jul 18 05:44:59 CDT 2013

((0,0), 120.97 cpm)
((0,1), 93.77 cpm)
((0,2), 108.32 cpm)
((0,3), 101.45 cpm)
((0,4), 112.87 cpm)
((0,5), 98.45 cpm)
((0,6), 119.32 cpm)
((0,7), 121.21 cpm)
…

Step 1: Making a heatmap data

- Start the Grunt Shell.
  - $ pig
- Load the radiation.csv data in HDFS.
  - $ raw = LOAD 'radiation.csv' USING PigStorage(',') AS (CapturedTime : chararray,Latitude:double,Longitude:double,Value:int,Unit:chararray);
- Step 1: Filter out CaptureTime.
  - $ step1 = FILTER raw BY CapturedTime is not null AND CapturedTime != 'null' AND CapturedTime != '';
- Step 2: select the rows of which unit is cpm.
  - $ step2 = FILTER step1 BY $4 MATCHES 'cpm';
- Step 3: Filter out the data based on the bounding box of Koriyama, Japan.
  - $ step3 = FILTER step2 BY $2 > 140.32564 AND $2 < 140.40993 AND $1 > 37.36191 AND $1 < 37.43678;

Step 1: Making a heatmap data

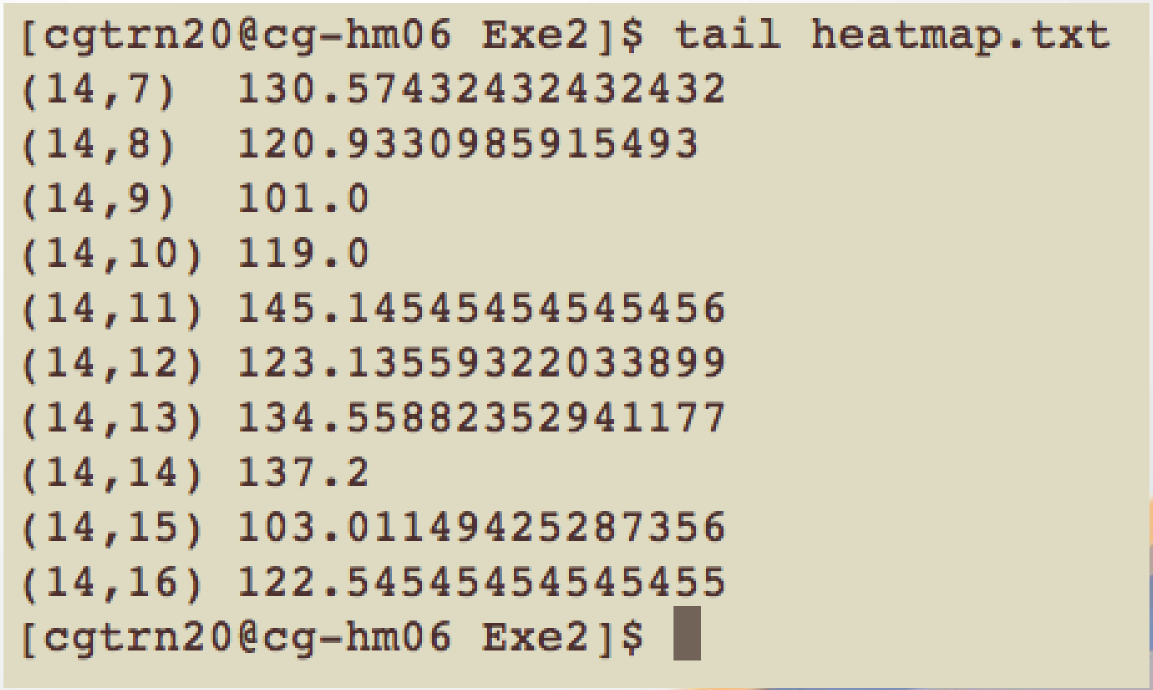
- Step 4: Select the rows of which CapturedTime is August.
  - $ step4 = FILTER step3 BY ($0 > '2013-08-01 00:00:00' AND $0 < '2013-09-01 00:00:00');
- Step 5: Generate new records to send down the pipeline to the next operator.
  - $ step5 = FOREACH step4 GENERATE $0,$1,$2,$3,$4;
- Step 6: Generate new records based on latitude and longitude key.
  - $ step6 = FOREACH step5 GENERATE $3,(int)(($2 - 140.32564)/0.005) AS logkey,(int)(($1 - 37.36191)/0.005) AS latkey;
- Step 7: Collect all records with the same latitude and longitude key.
  - $ step6_group = GROUP step6 BY (latkey,logkey);

Step 1: Making a heatmap data

- Step 8: Calculate the average of radiation levels per each cell.
  - $ step6_avg = FOREACH step6_group GENERATE group, AVG(step6.Value);
- Step 9: Sort data based on the group.
  - $ result = ORDER step6_avg BY group;
- Step 10: Send the output to a folder in HDFS.
  - $ Store result into 'heatmap_output';
- Step 11: Log out the Grunt shell
  - $ quit (or by Ctr + c)

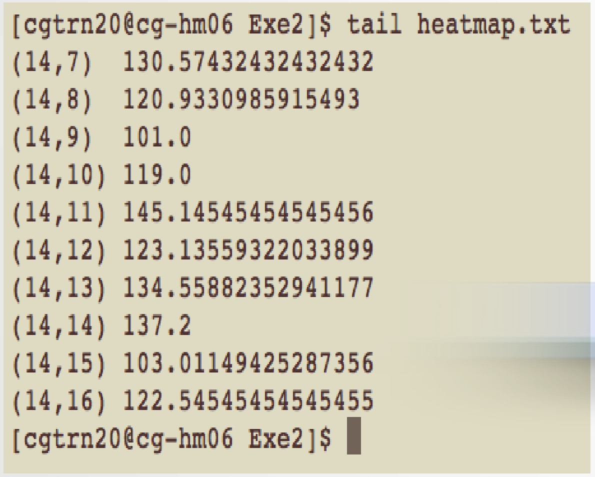
Step 2: Making a heatmap

- Step 13: Take a source directory as input and concatenate files into the destination local file
  - $ hadoop fs -getmerge heatmap_output ~/lecture7/demo1/heatmap.txt
- Check the content of heatmap.txt
  - $ tail heatmap.txt
  - python demo1.py

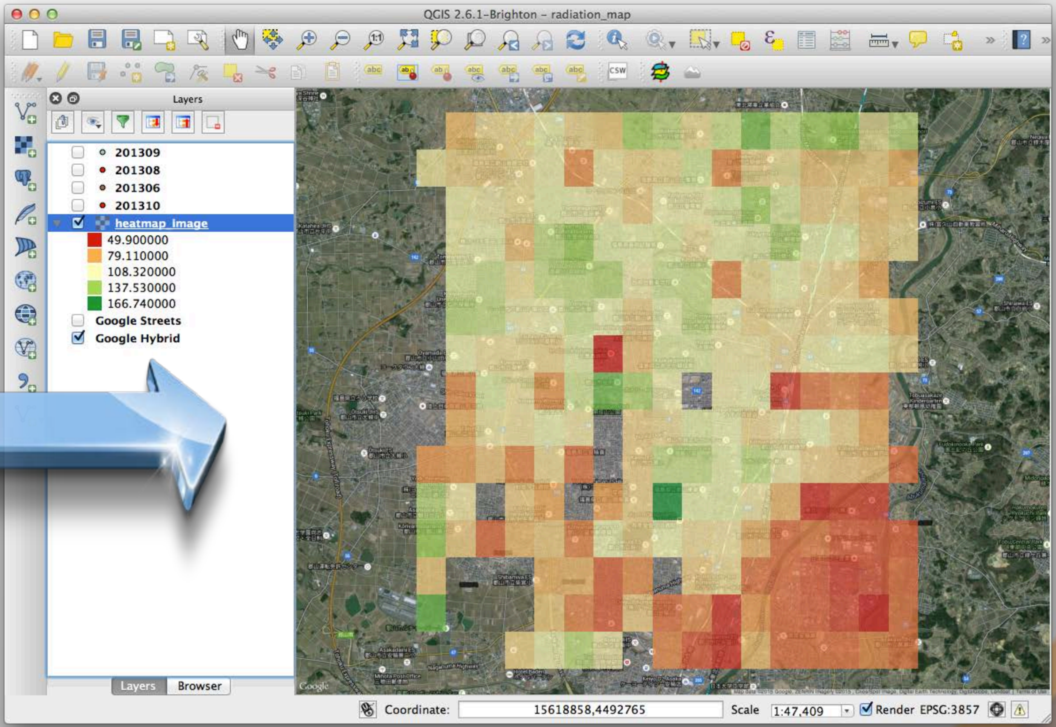


```
[cgtrn20@cg-hm06 Exe2]$ tail heatmap.txt
(14,7)  130.57432432432432
(14,8)  120.9330985915493
(14,9)  101.0
(14,10) 119.0
(14,11) 145.14545454545456
(14,12) 123.13559322033899
(14,13) 134.55882352941177
(14,14) 137.2
(14,15) 103.01149425287356
(14,16) 122.54545454545455
[cgtrn20@cg-hm06 Exe2]$
```

Visualize the generated heatmap



```
[cgtrn20@cg-hm06 Exe2]$ tail heatmap.txt
(14,7)  130.57432432432432
(14,8)  120.9330985915493
(14,9)  101.0
(14,10) 119.0
(14,11) 145.14545454545456
(14,12) 123.13559322033899
(14,13) 134.55882352941177
(14,14) 137.2
(14,15) 103.01149425287356
(14,16) 122.54545454545455
[cgtrn20@cg-hm06 Exe2]$
```



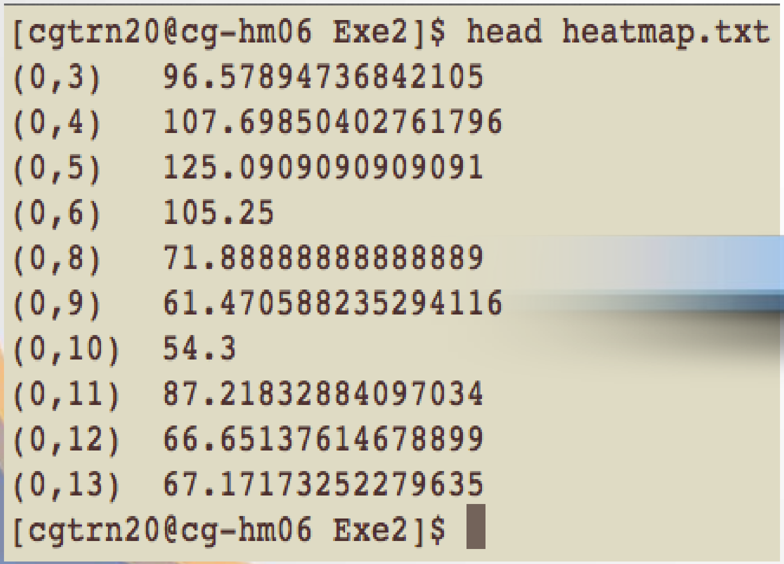
Analyzing data with Pig UDF using Python

Python in ROGER

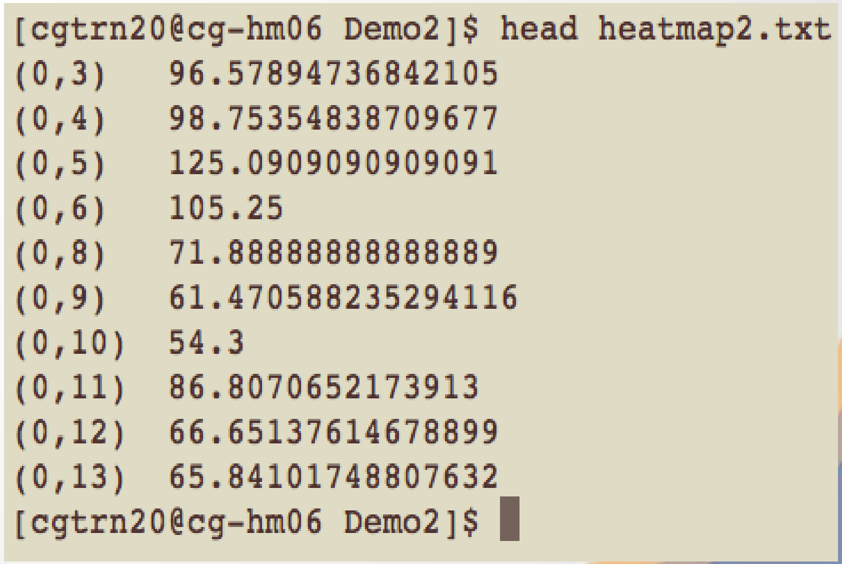
- There are multiple versions of Python installed in ROGER
  - List all the available Python environment in ROGER
- In particular, the Python environment comes with Anaconda is used in this course
  - scipy, numpy, Pandas, Sciki-learn, gdal-python, etc.
  - IPython
  - IPython notebook

Case 2: Generating a heatmap after removing outliers using Pig UDF

- Objective:
  - Make a heatmap image file (Ascii format) after removing outliers per each cell.
  - Compare the result with the one from the previous exercise.



```
[cgtrn20@cg-hm06 Exe2]$ head heatmap.txt
(0,3)   96.57894736842105
(0,4)   107.69850402761796
(0,5)   125.0909090909091
(0,6)   105.25
(0,8)   71.88888888888889
(0,9)   61.470588235294116
(0,10)  54.3
(0,11)  87.21832884097034
(0,12)  66.65137614678899
(0,13)  67.17173252279635
[cgtrn20@cg-hm06 Exe2]$
```



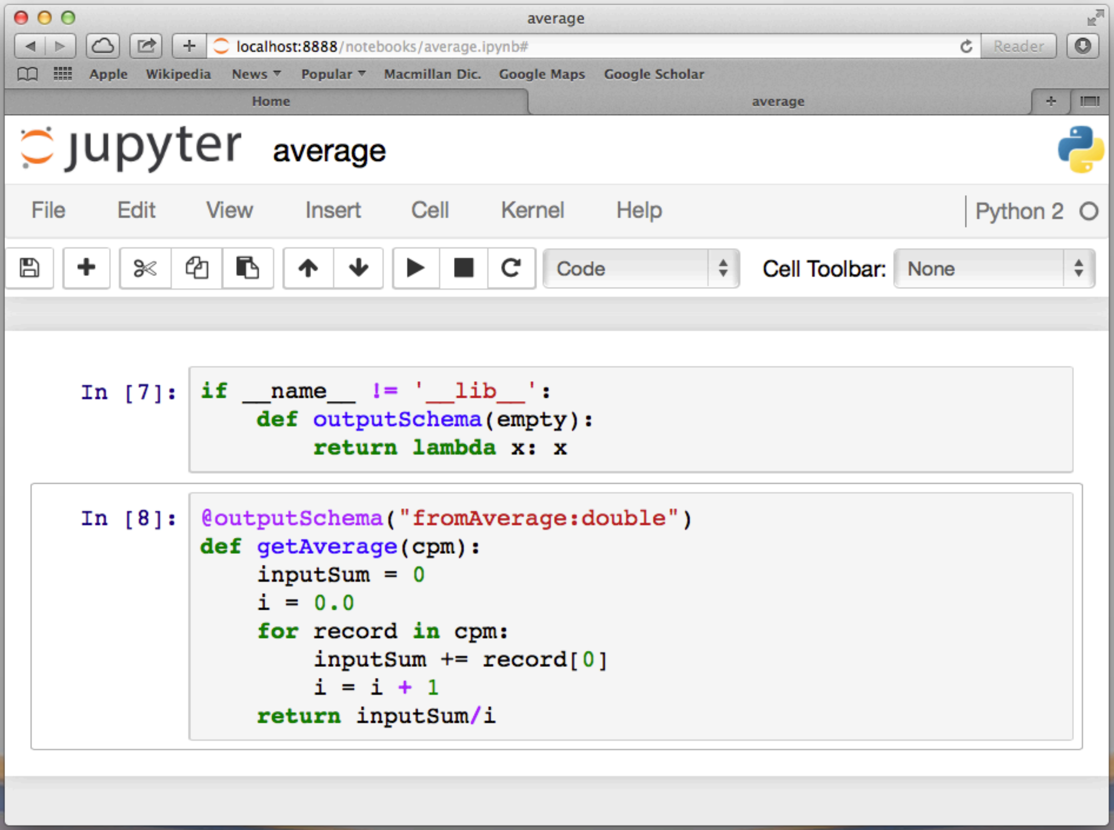
```
[cgtrn20@cg-hm06 Demo2]$ head heatmap2.txt
(0,3)   96.57894736842105
(0,4)   98.75354838709677
(0,5)   125.0909090909091
(0,6)   105.25
(0,8)   71.88888888888889
(0,9)   61.470588235294116
(0,10)  54.3
(0,11)  86.8070652173913
(0,12)  66.65137614678899
(0,13)  65.84101748807632
[cgtrn20@cg-hm06 Demo2]$
```

Pig User Defined Functions (UDFs)

- Alternatives to Hadoop streaming for running Python on a cluster
  - Pig User Defined Functions (UDFs): Invoking Python code from within a Pig script
  - Pig Streaming: Using the Pig STREAM command
- Combine Pig's operators with their own or others' code via UDFs
- Pig provides extensive support for user defined functions (UDFs) as a way to specify custom processing
  - Java
  - Jython: a Python implementation that runs on a JVM
  - Python (Python UDFs do not work on Hadoop 2.x)
  - Javascript, Ruby and Groovy (experimental)

Writing a Pig UDF in Python

- A UDF written in Python can be any Python function.
- Annotate the function using @outputSchema so Pig can determine what the data type of the return value is.


average
localhost:8888/notebooks/average.ipynb#
Apple Wikipedia News Popular Macmillan Dic. Google Maps Google Scholar
Home
average
jupyter average
File Edit View Insert Cell Kernel Help
Python 2
Code
Cell Toolbar: None

```
In [7]: if __name__ != '__lib__':
            def outputSchema(empty):
                return lambda x: x
```

```
In [8]: @outputSchema("fromAverage:double")
        def getAverage(cpm):
            inputSum = 0
            i = 0.0
            for record in cpm:
                inputSum += record[0]
                i = i + 1
            return inputSum/i
```

Invoking a UDF

- Before invoking a Python UDF, we need to register the text file that contains the function definition.
  - register 'udfs.py' using jython as udfs;
  - A = LOAD ‘work’ AS (department:chararray,name:chararray,salary:double);
  - B = GROUP A BY department;
  - C = FOREACH B GENERATE group, udfs.getAverage(A.Value) ;
- By incorporating the UDF in the Pig script, we can take advantage of Pig’s ability to manipulate and group our big data, while also taking advantage of Python’s powerful but simple processing and logic.

Invoking a UDF

- Work on the directory where you have the scripts ready
  - cd ~/lecture7/demo2
- Run the pig script and see the results
  - What are the precedures?

Case 3: Implementing K-Means clustering on Hadoop for 311 Data

- Objective:
  - Run a Pig script that streams data through a Python script.
  - Use Scikit-Learn to generate a K-Means clustering for 311 Data.

- Before you begin:
  - You need to load module anaconda, which contains the k-mean clustering library you need
    - module load ?
  - Work on the directory where you have the scripts
    - cd ~/lecture7/demo3

Pig STREAM Command

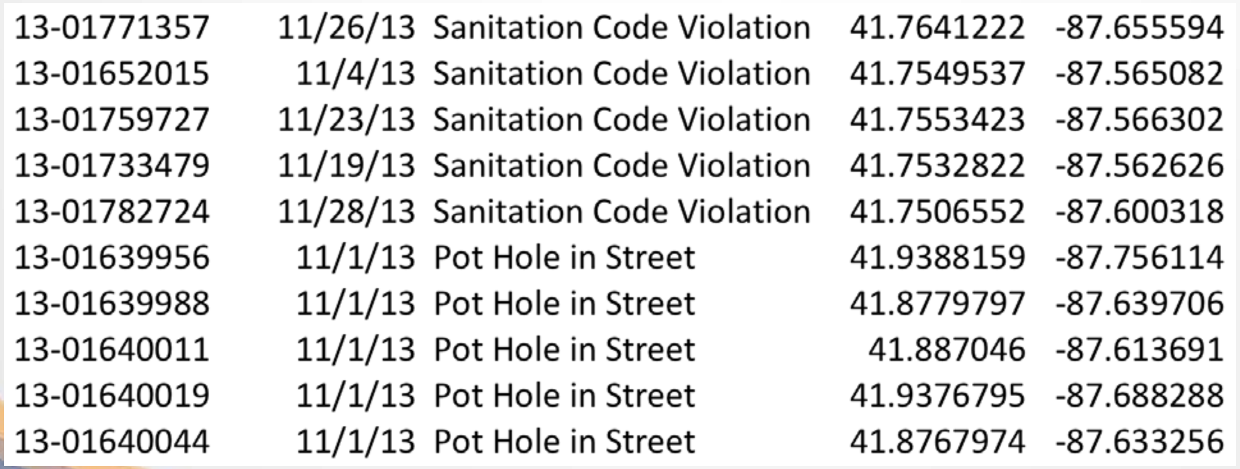
- The Pig STREAM command sends data to an external script or program. The syntax looks like:
    - alias = STREAM alias [, alias …] THROUGH {`command` | cmd_alias} [AS schema] ;
    - E.g., x = LOAD ’311_data';

      result = STREAM x THROUGH `kmeans`;

- Defining a streaming alias
    - Use the DEFINE command to assign an alias to a streaming command.
    - **DEFINE kmeans `kmeans311.py` SHIP (‘kmeans311.py’)**
    - Specify files that are needed on the worker nodes when using Pig streaming.

311 Dataset

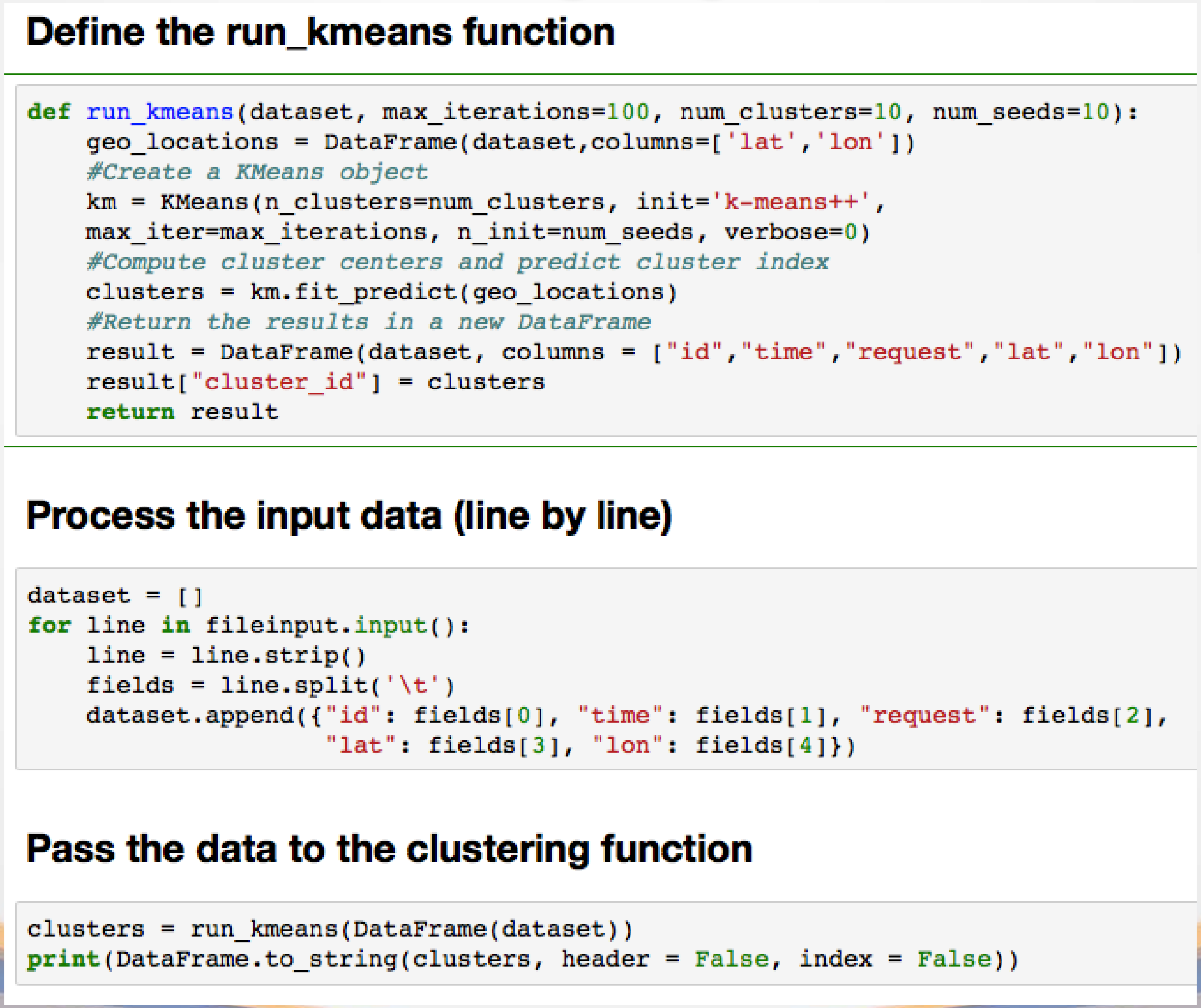
- 311 is for non-emergency services.
- Improved customer service for city departments and governmental agencies
- Service requests:
  - Graffiti Removal, Vacant and Abandoned Buildings, Pot Holes, etc.
- ID, Date, Request service, Latitude, Longitude



13-01771357	11/26/13	Sanitation Code Violation	41.7641222	-87.655594
13-01652015	11/4/13	Sanitation Code Violation	41.7549537	-87.565082
13-01759727	11/23/13	Sanitation Code Violation	41.7553423	-87.566302
13-01733479	11/19/13	Sanitation Code Violation	41.7532822	-87.562626
13-01782724	11/28/13	Sanitation Code Violation	41.7506552	-87.600318
13-01639956	11/1/13	Pot Hole in Street	41.9388159	-87.756114
13-01639988	11/1/13	Pot Hole in Street	41.8779797	-87.639706
13-01640011	11/1/13	Pot Hole in Street	41.887046	-87.613691
13-01640019	11/1/13	Pot Hole in Street	41.9376795	-87.688288
13-01640044	11/1/13	Pot Hole in Street	41.8767974	-87.633256

311 City Services – http://www.cityofchicago.org/city/en/depts/311.html

Python code to implement the K-Means clustering algorithm



Define the run_kmeans function

```python
def run_kmeans(dataset, max_iterations=100, num_clusters=10, num_seeds=10):
    geo_locations = DataFrame(dataset,columns=['lat','lon'])
    #Create a KMeans object
    km = KMeans(n_clusters=num_clusters, init='k-means++',
    max_iter=max_iterations, n_init=num_seeds, verbose=0)
    #Compute cluster centers and predict cluster index
    clusters = km.fit_predict(geo_locations)
    #Return the results in a new DataFrame
    result = DataFrame(dataset, columns = ["id","time","request","lat","lon"])
    result["cluster_id"] = clusters
    return result
```

Process the input data (line by line)

```python
dataset = []
for line in fileinput.input():
    line = line.strip()
    fields = line.split('\t')
    dataset.append({"id": fields[0], "time": fields[1], "request": fields[2],
                    "lat": fields[3], "lon": fields[4]})
```

Pass the data to the clustering function

```python
clusters = run_kmeans(DataFrame(dataset))
print(DataFrame.to_string(clusters, header = False, index = False))
```

Stream the data through a clustering algorithm using Pig streaming

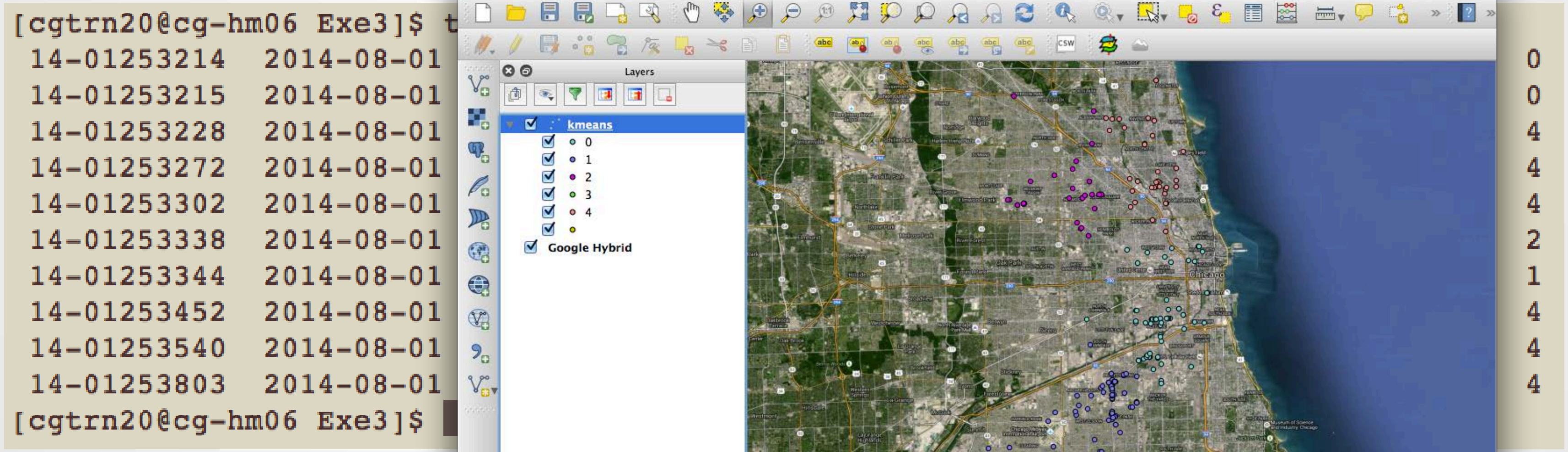
- define kmeans311 `/sw/anaconda/bin/python kmeans311.py` ship ('kmeans311.py');
- raw = LOAD ’cybergis/311_data.csv' USING PigStorage(',') AS (id:chararray,time:chararray,request:chararray,Lat:double,Lon:double);
- step0 = FILTER raw BY time != 'null' AND time != '';
- step1 = FILTER step0 BY $0 is not null;
- step2 = FILTER step1 BY $1 is not null;
- step3 = FILTER step2 BY $2 is not null and $2 MATCHES 'Alley Light Out’;
- step4 = FOREACH step3 GENERATE $0,$1,$2,$3,$4;
- result = stream step4 through kmeans311;
- store result into 'kmeans_output';

K-Means Clustering

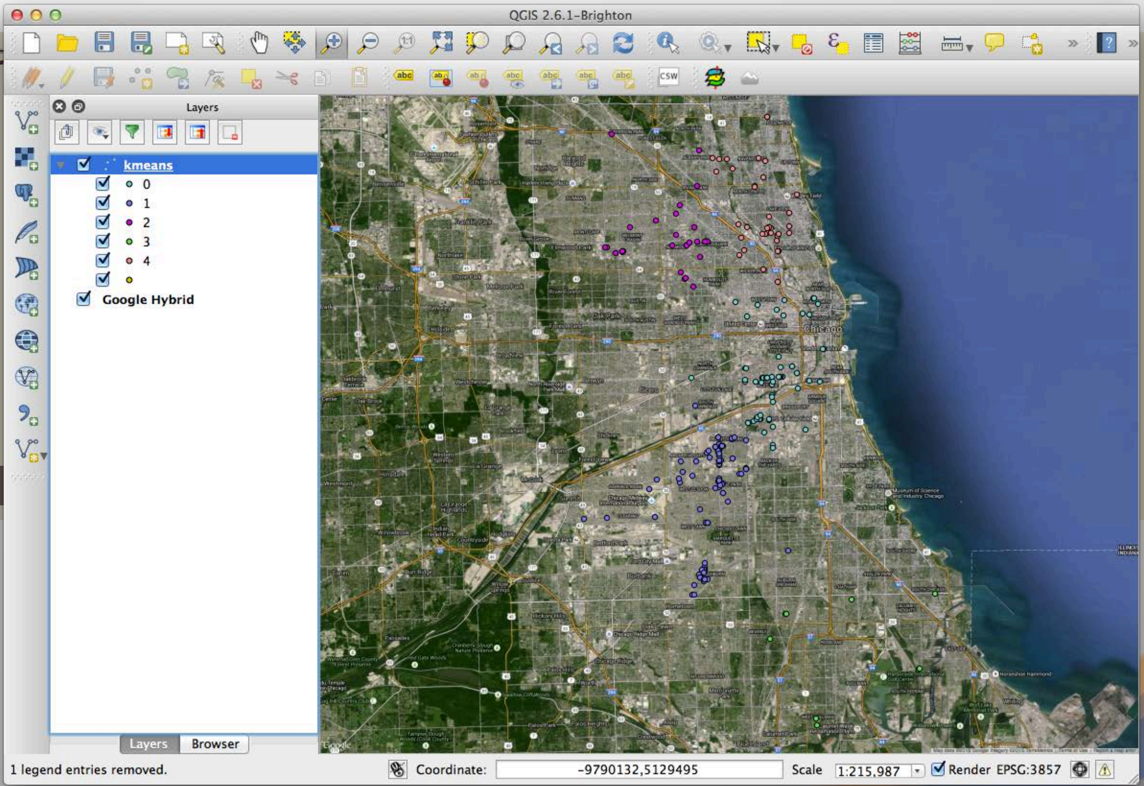
- Step 1: Put the data in HDFS
  - $hdfs dfs -copyFromLocal 311_data.csv
- Step 2: Run the Pig Script
  - $pig kmeans311.pig
  - Save the result to the local directory
  - $hdfs dfs -getmerge kmeans_output kmeans.csv
  - Visualize the results as points in ArcGIS or QGIS
    - Take advantage of MobaXterm

K-Means Clustering

- Step 3: View the result
  - **$tail kmeans.csv**
  - K-Means clustering for Graffiti removal requests from 01/15/2015 to 08/01/2014.



```
[cgtrn20@cg-hm06 Exe3]$ t
 14-01253214  2014-08-01                                                0
 14-01253215  2014-08-01                                                0
 14-01253228  2014-08-01                                                4
 14-01253272  2014-08-01                                                4
 14-01253302  2014-08-01                                                4
 14-01253338  2014-08-01                                                2
 14-01253344  2014-08-01                                                1
 14-01253452  2014-08-01                                                4
 14-01253540  2014-08-01                                                4
 14-01253803  2014-08-01                                                4
[cgtrn20@cg-hm06 Exe3]$
```



1 legend entries removed. Coordinate: -9790132,5129495 Scale 1:215,987 Render EPSG:3857