2011-2014 CALIFORNIA PAYROLL ANALYSIS

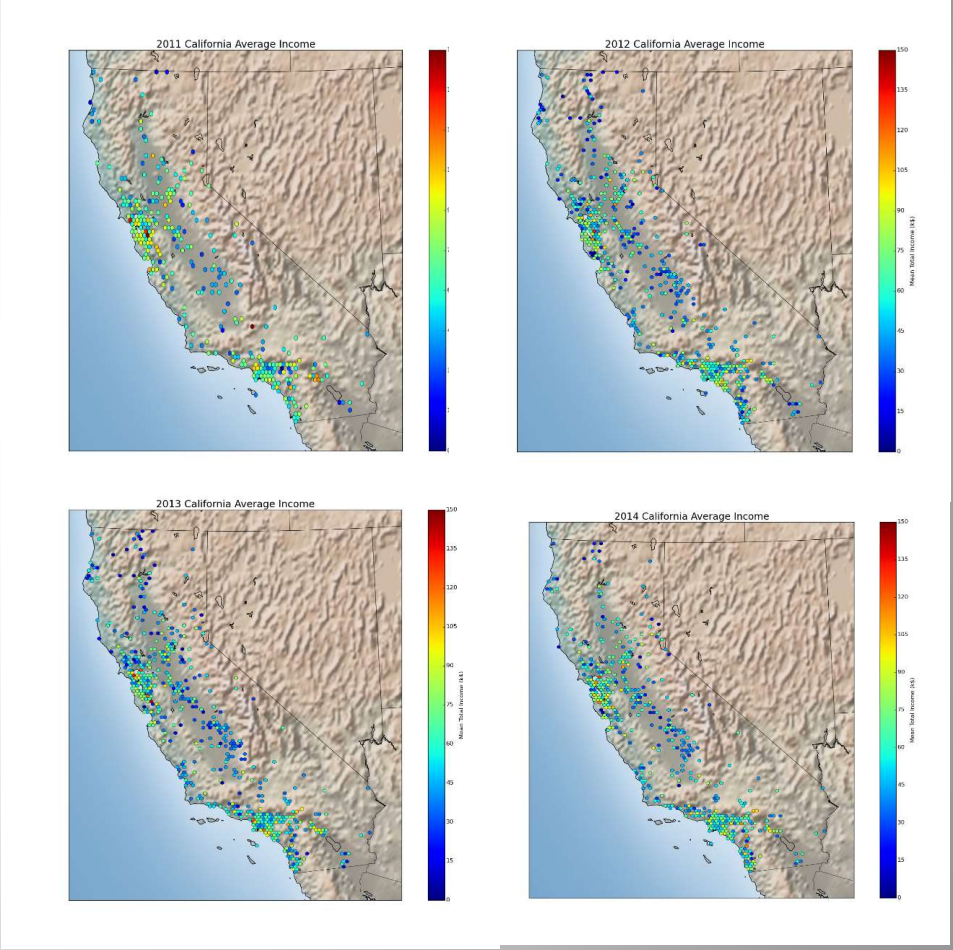
CSC550 CAPSTONE
JINZHONG.ZHANG, XU.LIAN
03/19/2016

AGENDA

- Project Description
- Data Description
- Views Over the Years
- Major Findings
- Technical Approach
- Summary of Work Experience

PROJECT DESCRIPTION

Based on the released California payroll data between 2011 to 2014, this project mainly focus on analyzing the trends of total income and the income geographic distribution

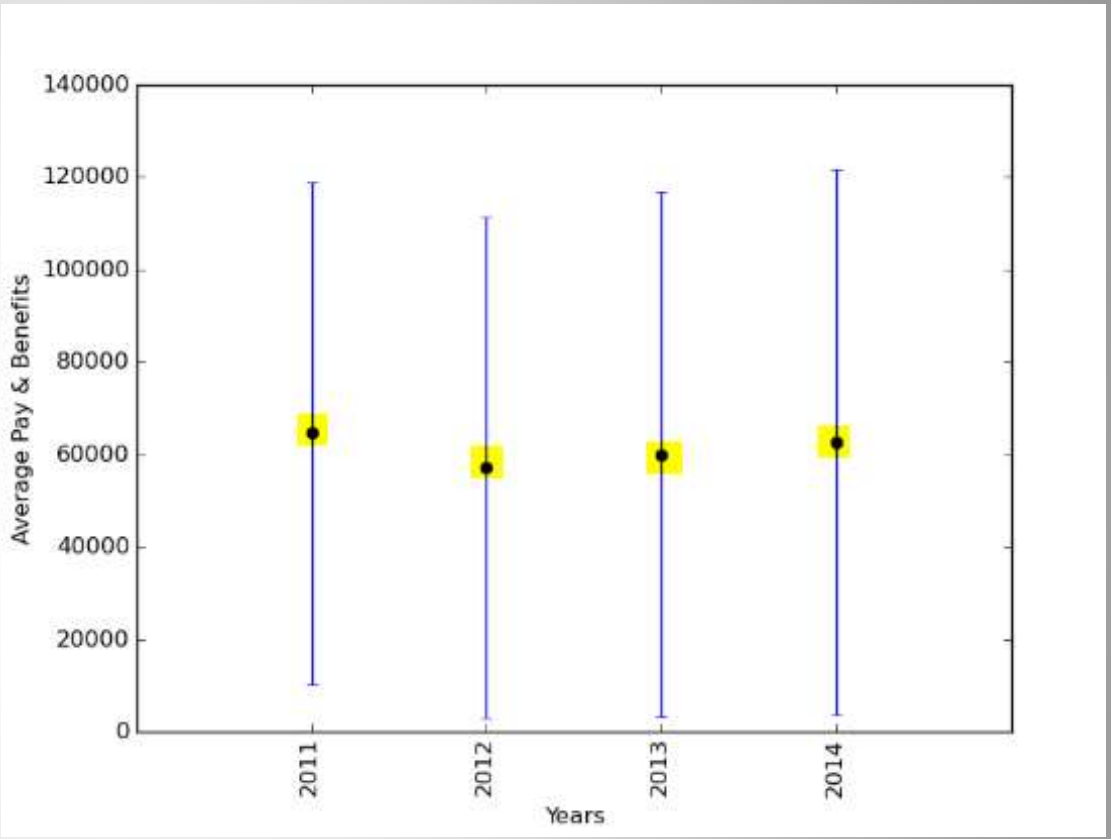


DATA DESCRIPTION

- Data Source: California payroll data set from 2011 to 2014
- Data Size and effective records
  - total: 960.6MB
  - 2011: ~1.2 million payroll records
  - 2012: ~2.0 million payroll records
  - 2013: ~2.3 million payroll records
  - 2014: ~2.4 million payroll records
- Effective locations analyzed: over 1200 cities
- Parallelized threads: 8-34 threads, according to data size
- Data Format (150MB to 250MB csv file for each year):
  "Employee Name", "Job Title", "Base Pay", "Overtime Pay", "Other Pay", "Benefits", "Total Pay", "Total Pay & Benefits", "Year", "Notes", "Agency (City name)"

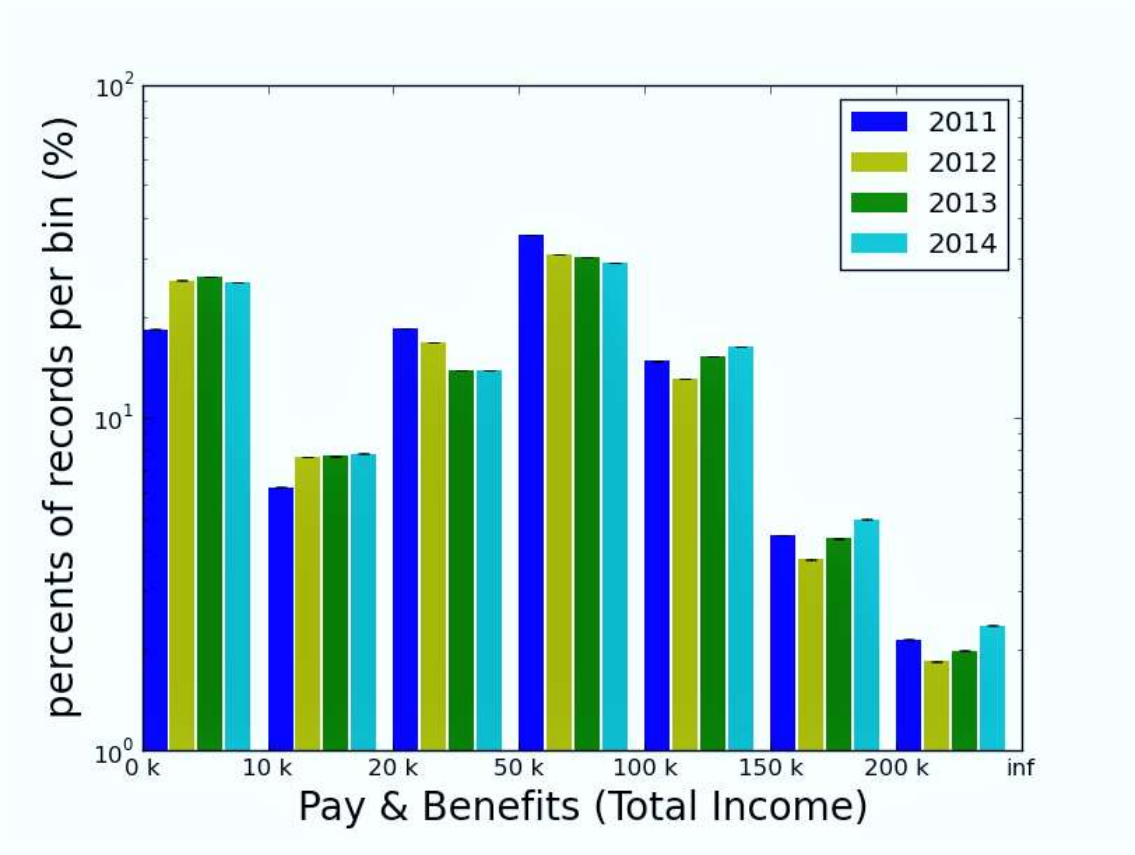
Views Over the Years

AVERAGE PAYROLL COMPARE



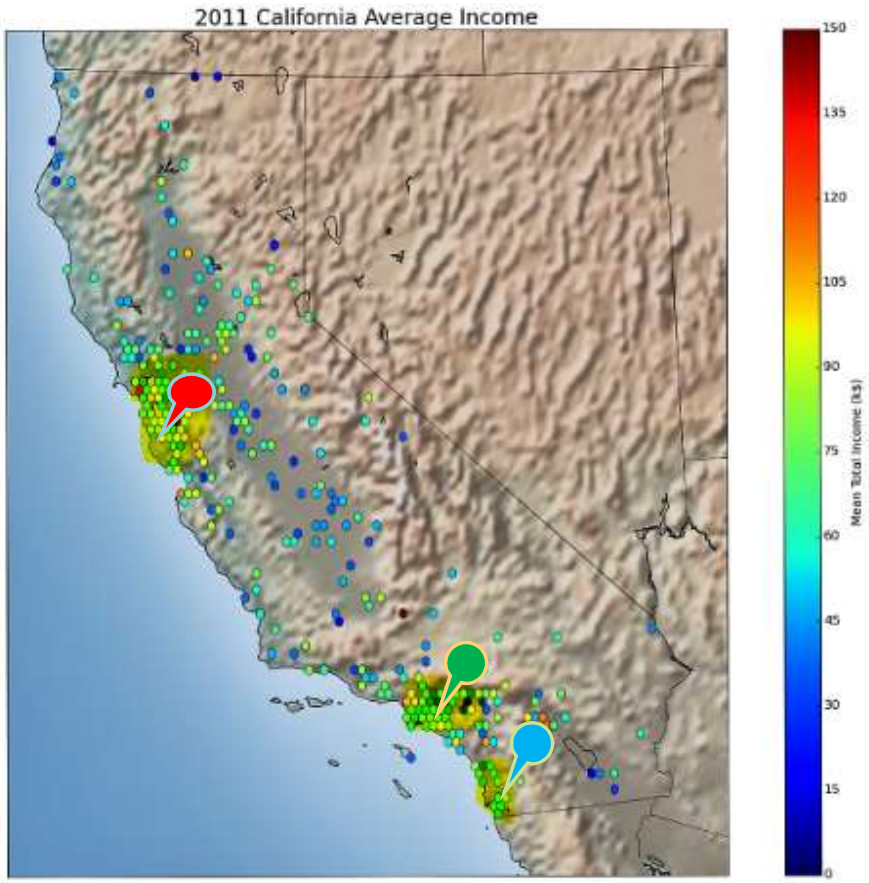
TOTAL INCOME COMPARISON AMONG YEARS

- Total Income trends among years
- Difference between average total income and medium total income



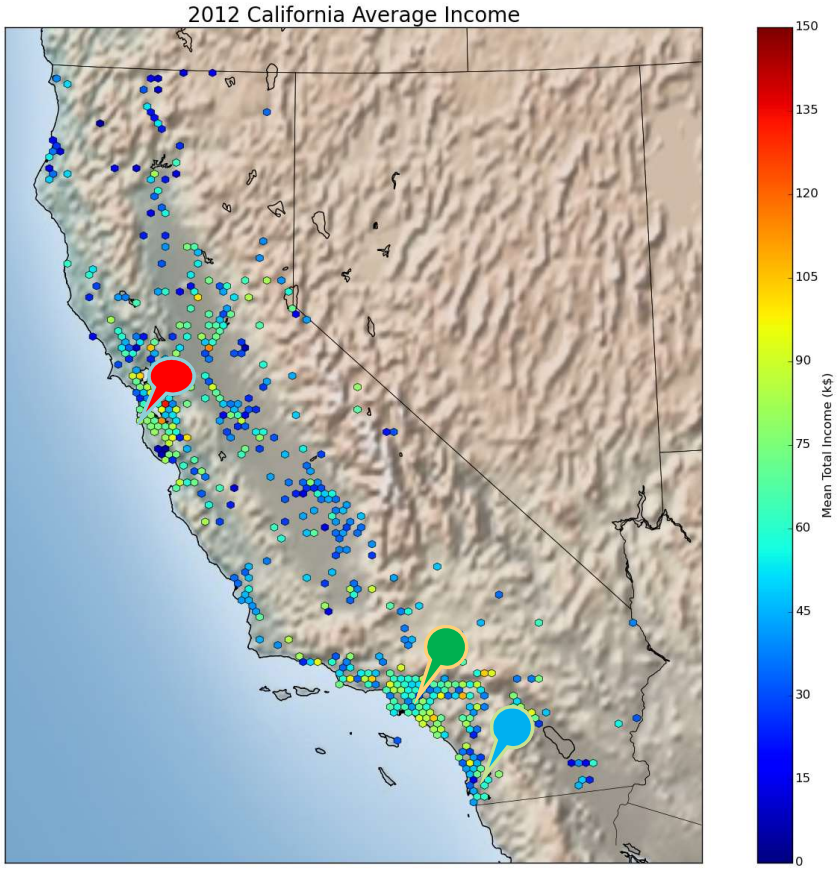
2011 CALIFORNIA AVERAGE INCOME

- San Francisco bay area
- Los Angeles
- San Diego



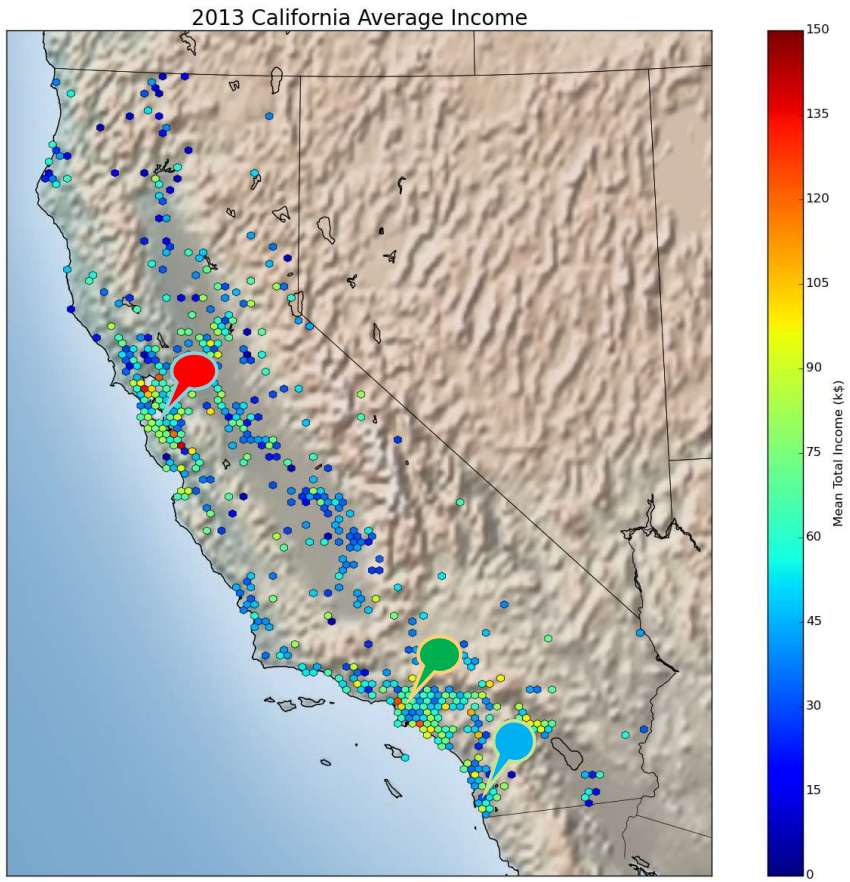
2012 CALIFORNIA AVERAGE INCOME

- San Francisco bay area
- Los Angeles
- San Diego



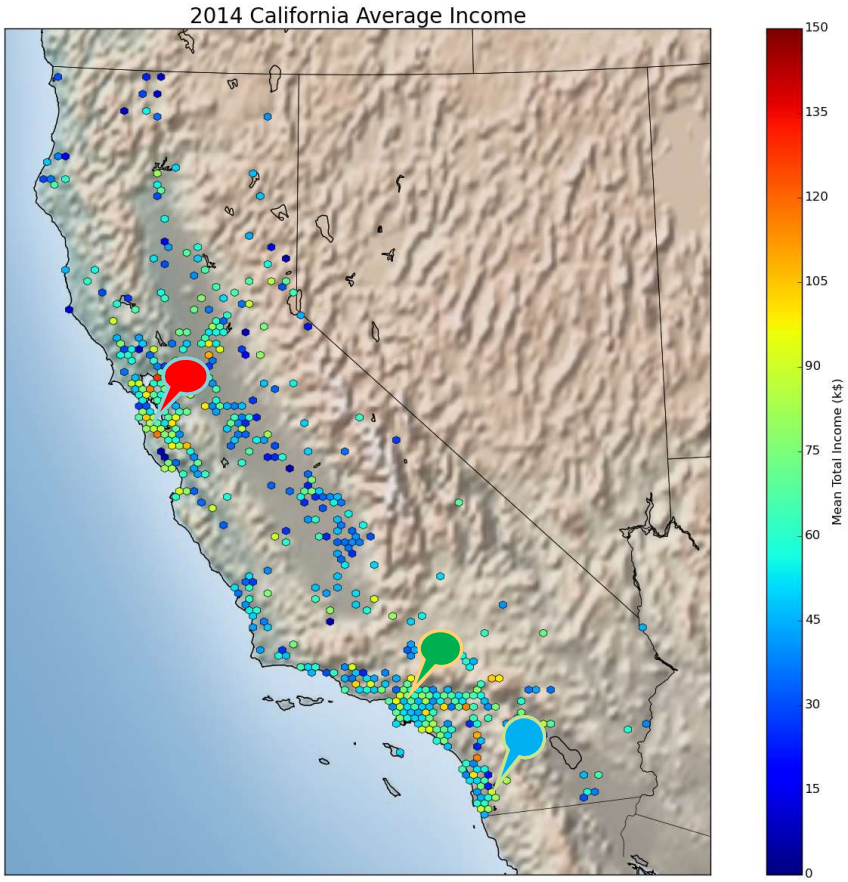
2013 CALIFORNIA AVERAGE INCOME

- San Francisco bay area
- Los Angeles
- San Diego



2014 CALIFORNIA AVERAGE INCOME

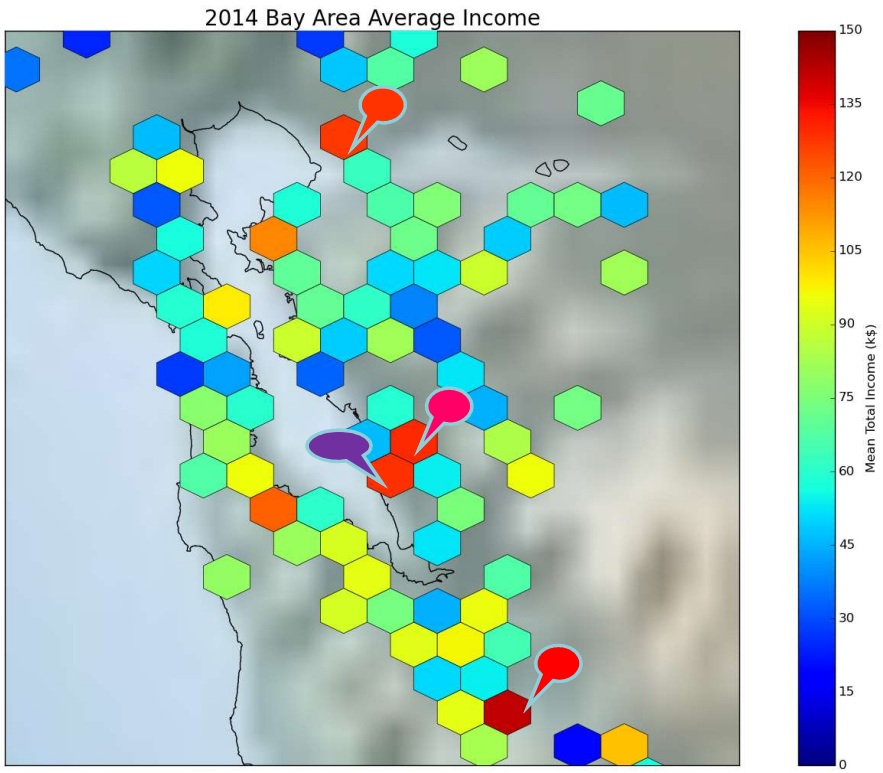
- San Francisco bay area
- Los Angeles
- San Diego



2014 SAN FRANCISCO BAY AREA AVERAGE INCOME

Relative high income cities:

- Oakland
- Fremont
- Newark
- San Jose



PERFORMANCE

MAKE HISTOGRAM & SCATTER PLOT

- All ~8 million records:

sec

- Compare to SQL:

>1 day?

HEXBIN PLOT ON MAP

- For one year ~2 million records:

sec

- Compare to SQL:

? ? ?

TECHNICAL APPROACH

- System requirements and software dependence
- Installation
- Implementations

SYSTEM REQUIREMENTS AND SOFTWARE DEPENDENCE

- Spark-1.6.0
- Python >= 2.7
- (optional, for single machine not necessary) Hadoop HDFS 2.0

- Python packages:
  - matplotlib (1.4.2) → python plot making library
  - basemap (1.0.7) → make plot on real map
  - geopy (1.11.0) → convert addresses or county names to (latitude, longitude)
  - http://download.osgeo.org/geos/geos-3.5.0.tar.bz2 → geopy dependence

- Cloudera VM is not supported.

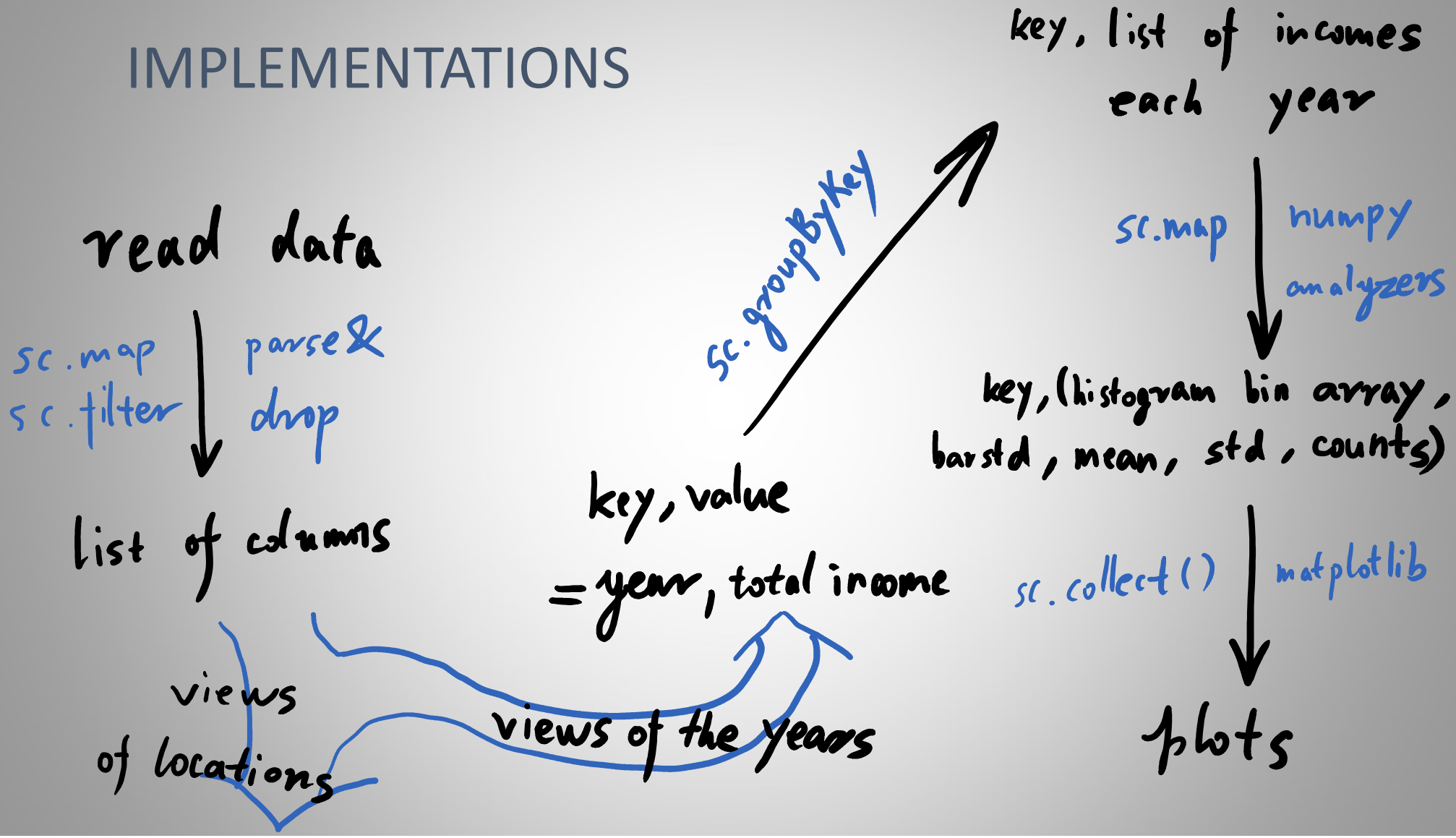
  This is because the Hue job monitor of Cloudera uses Python 2.6 . geo & map is not supported by Python 2.6. Python 2.6 is no longer supported by the Python team.

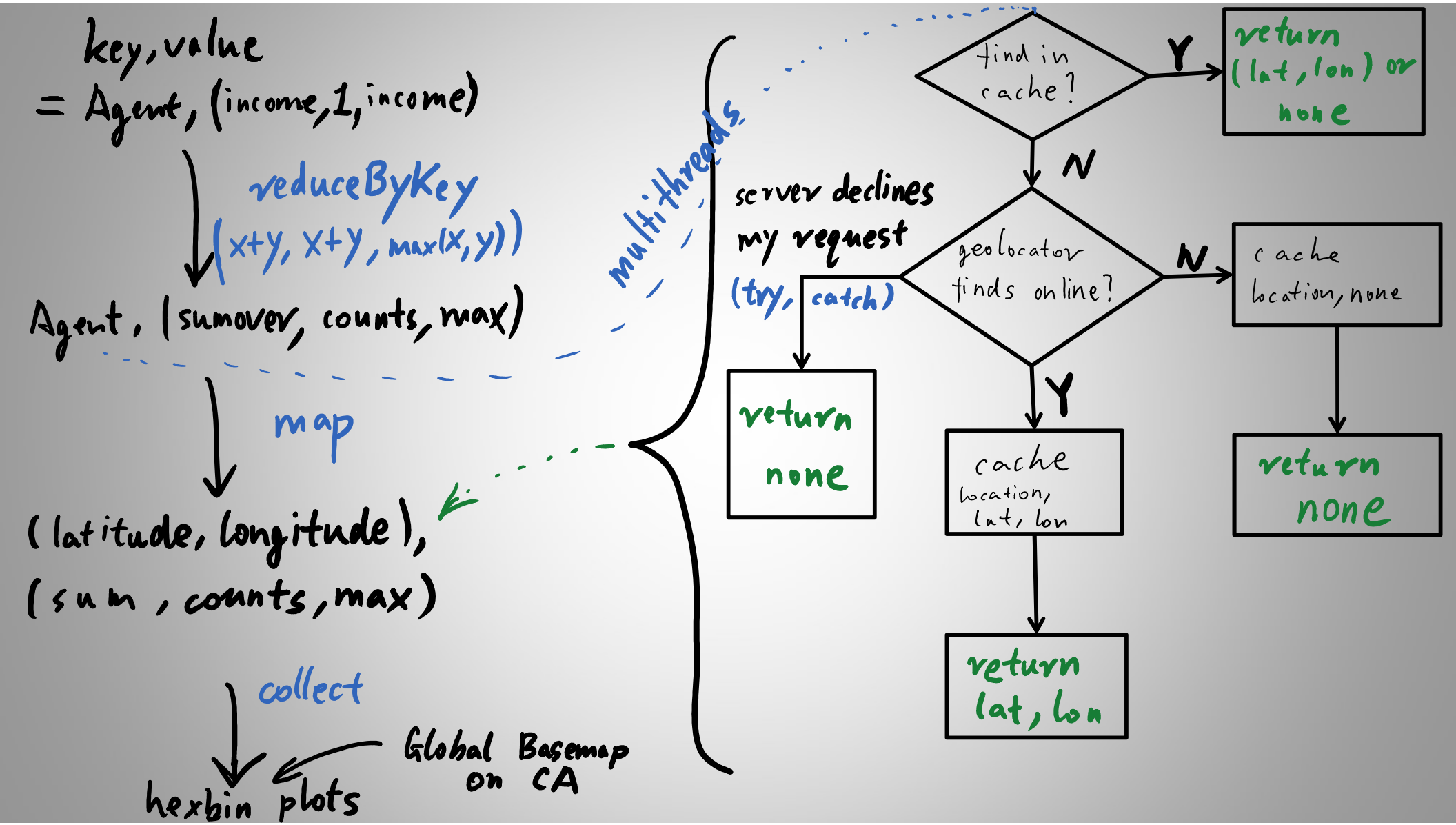
INSTALLATION

- Check versions:
  python --version
  pip list | grep ‘basemap\|matplot\|geopy’

- Installation:
  sudo pip install geopy
  sudo pip install matplotlib
  sudo pip install basemap
  wget http://download.osgeo.org/geos/geos-3.5.0.tar.bz2
  tar -xvf geos-3.5.0.tar.bz2
  cd geos-3.5.0
  ./configure
  make
  sudo make install

IMPLEMENTATIONS




key, value
= Agent, (income, 1, income)
reduceByKey
(x+y, x+y, max(x,y))
Agent, (sumover, counts, max)
map
(latitude, longitude),
(sum, counts, max)
collect
hexbin plots
Global Basemap on CA
multithreads
find in cache?
Y
return (lat, lon) or none
N
geolocator finds online?
N
cache location, none
return none
Y
cache location, lat, lon
return lat, lon
server declines my request
(try, catch)
return none

SUMMARY OF WORK EXPERIENCE parsing

- some cells in csv include “,” inside “”
  csv.reader(StringIO(line),strict=True, skipinitialspace = True, quoting = csv.QUOTE_ALL, quotechar = '"', delimiter = ',', lineterminator = '\n')
- csvreader cannot handle the cases that “\n” is inside the cell, because spark will split it into two lines
  Use a buffer to let spark remember the previous record. However, this fix is deprecated, because in parallelized threads, the content in the buffer may belong to another thread. It is better to throw bad data away.
- none utf-8 code inside the original data file. The python will raise error.
  Fix: line=line.encode('utf-8','ignore')
- Experience: found a good regexp to recognize all kinds of numbers (float,int, scientific) from online
  ‘^[-+]?[0-9]*\.?[0-9]+(?:[eE][-+]?[0-9]+)?$

SUMMARY OF WORK EXPERIENCE

1. Spark filter takes a long to run.
   Improvement: play with keys but not filters to reuse data.
   E.g. groupByKey → map(histogram_maker)→collect() the results

2. For online services. Now most of them do not like distributed computing requests
   It is slow for large data. When spark uses 20-30 threads, the request are too many so that the remote server will block my IP for a certain amount of time.

   Fix: use a local cache, a lookup table to same the relation between the city names and coordinates.

→

Nowadays, many servers are NOT ready for distributed computing. They are NOT able to distinguish between distributed computing requests and attacks. Their load balancers still need to be improved.

→

Lots of opportunities