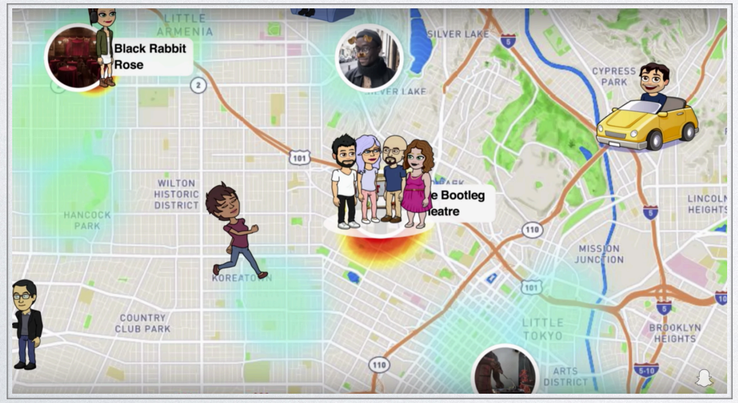
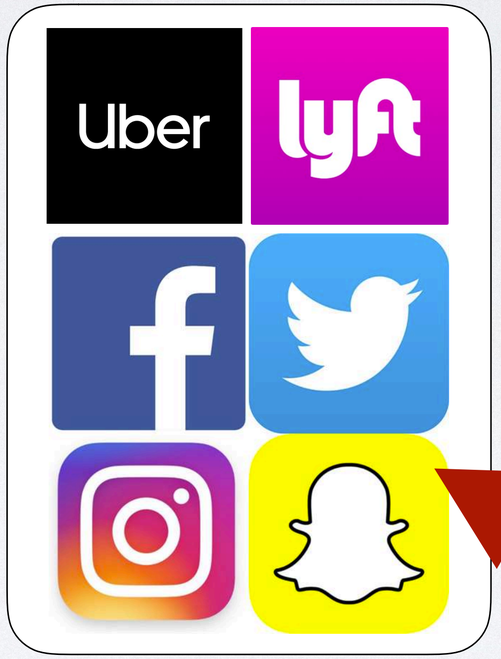
Spatial Data Wrangling With GeoSpark

Presented by:
Jia Yu
Mohamed Sarwat

Geospatial Data

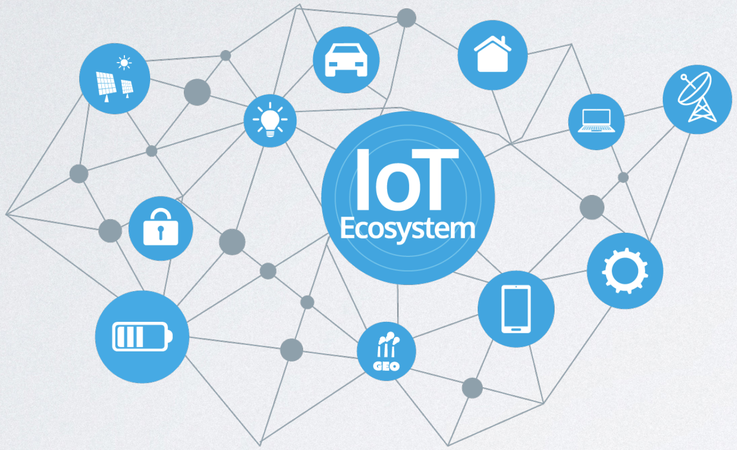
- Mobile devices - 4.68 billion in 2019

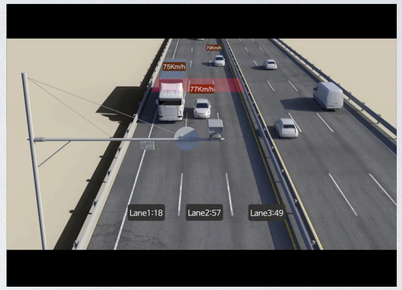




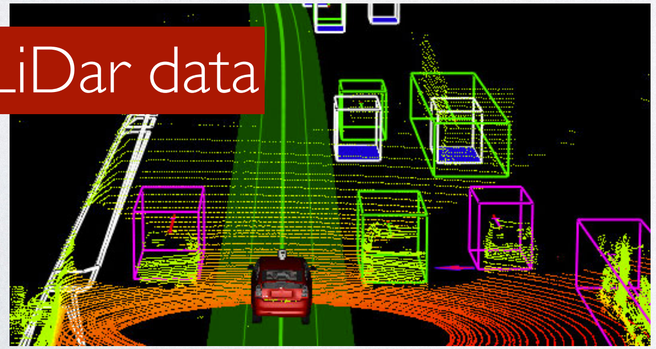
Geospatial Data

- IoT sensors in Smart City: 7 billion in 2019





Massive LiDar data

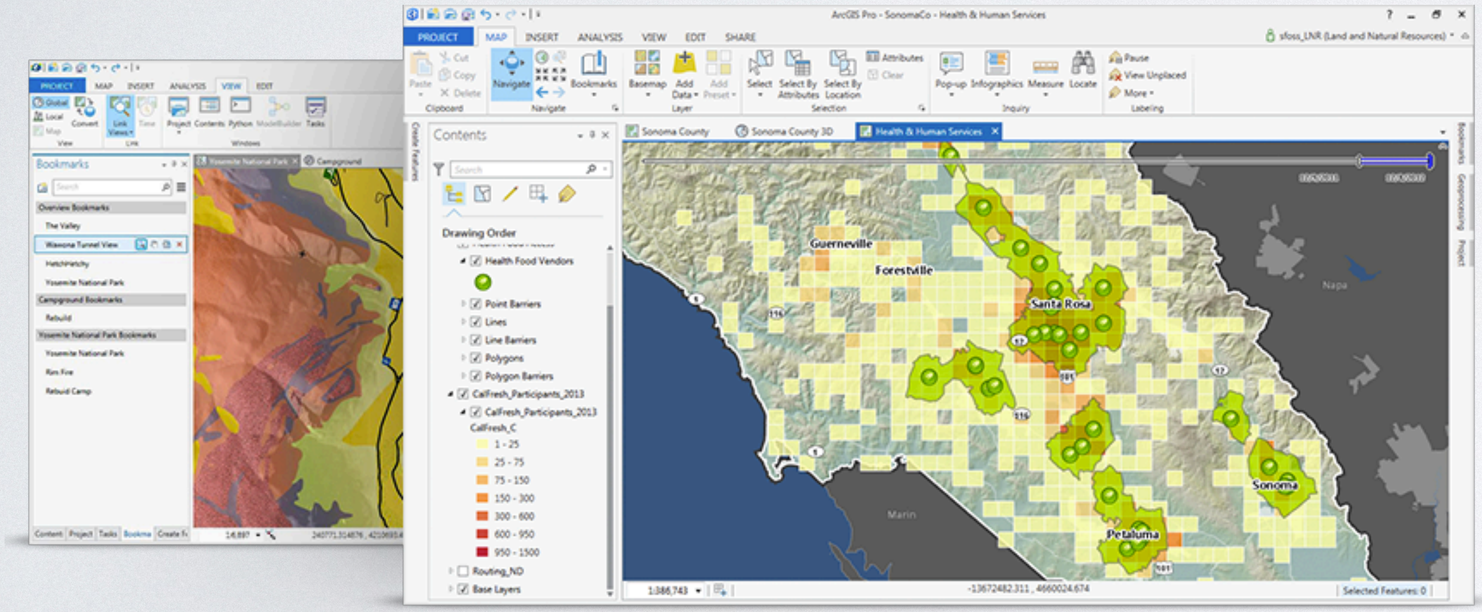


1.5 billion taxi trips

NYC
Taxi & Limousine Commission

Geospatial Data Frameworks

- Classic - single machine DBMS or GIS tools

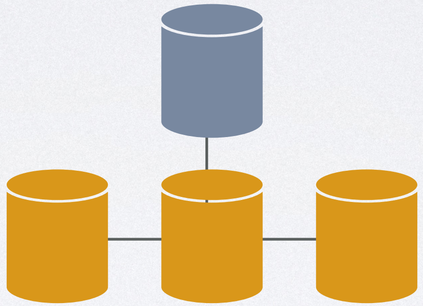


ArcGIS

QGIS

PostgreSQL + PostGIS

Cluster (Distributed) Computing Approaches



Manage Spatial Data in Spark?

I want to manage spatial data in Spark!

Not that easy!

- No spatial data type support
- No spatial index
- No spatial query

GeoSpark

http://datasystemslab.github.io/GeoSpark/

Google “GeoSpark ASU”

Arizona State University

mobike 摩拜单车

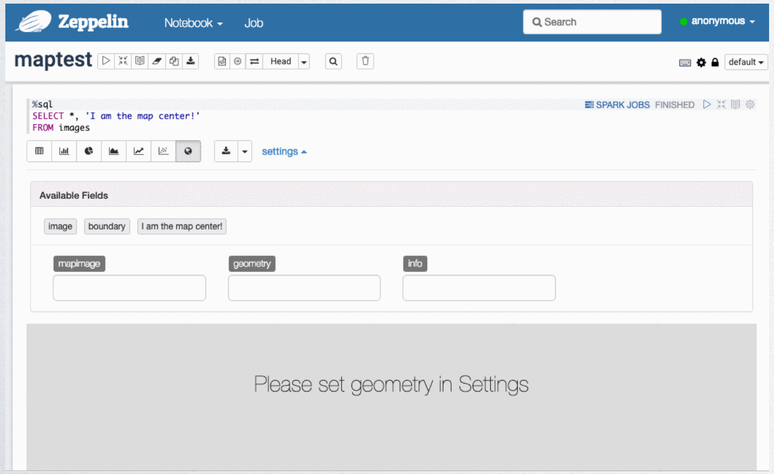
GYANA

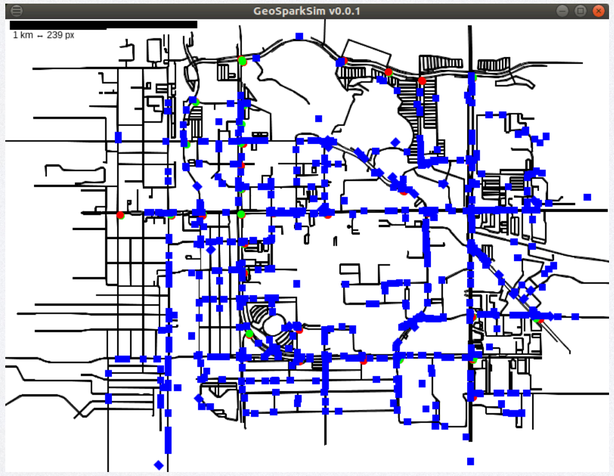
Blue DME

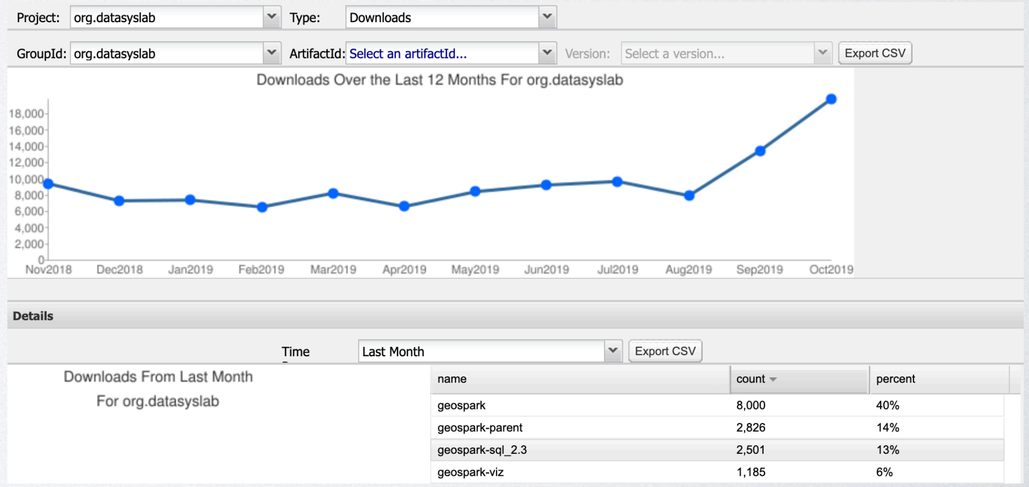
In production!

20K monthly downloads

- Spatial RDD, SQL, DataFrame on Apache Spark
- Distributed map visualization (Viz SQL)
- Integrated with Apache Zeppelin
- Built-in large-scale traffic simulator







Arizona State University

Outline

Instruction

Development

APACHE Spark™

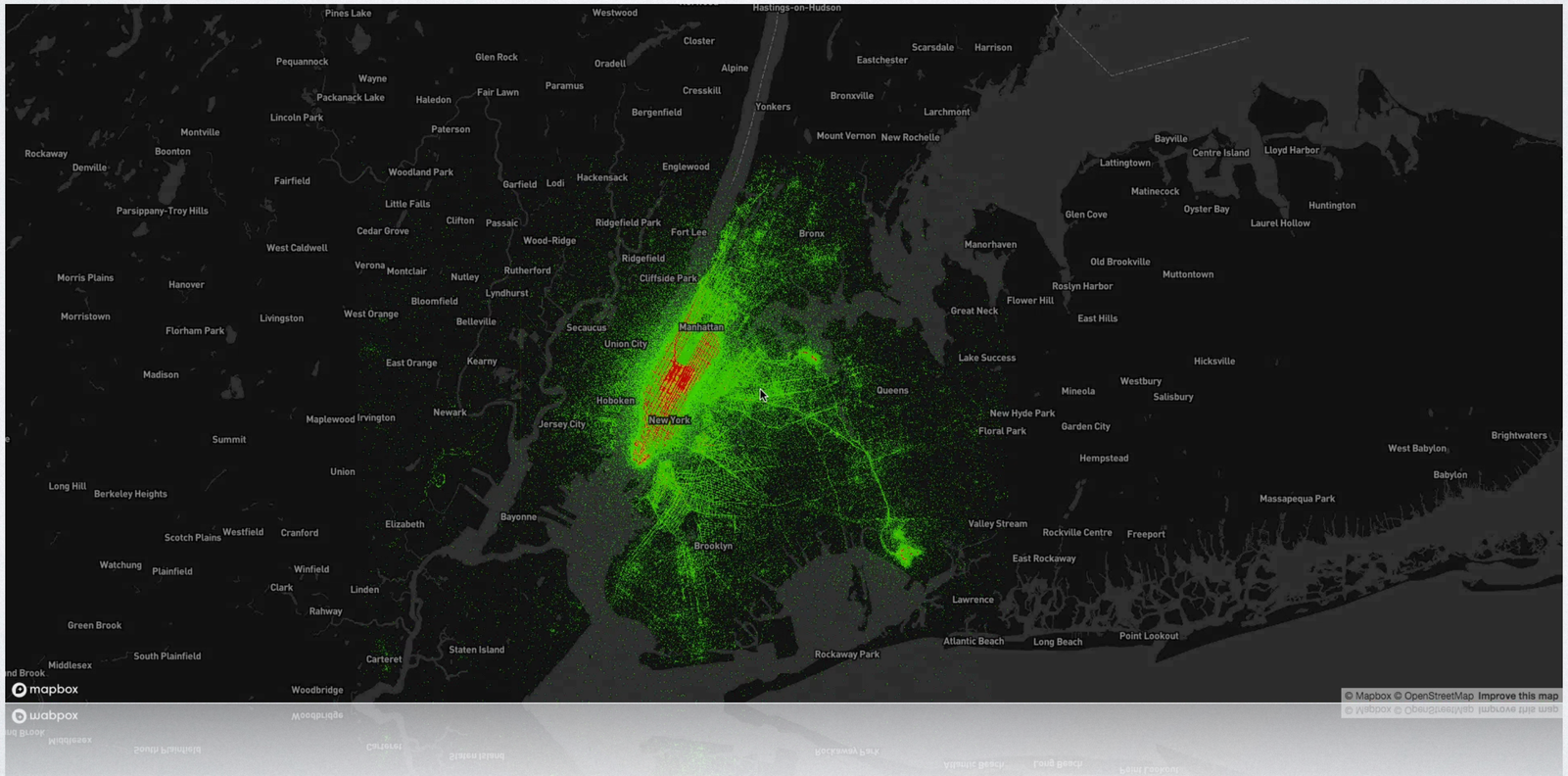
Manage Spatial Data

+ APACHE Spark™

- Spatial data partitioning
- Spatial indexing
- Spatial queries
- Optimization
- Language, spatial object support

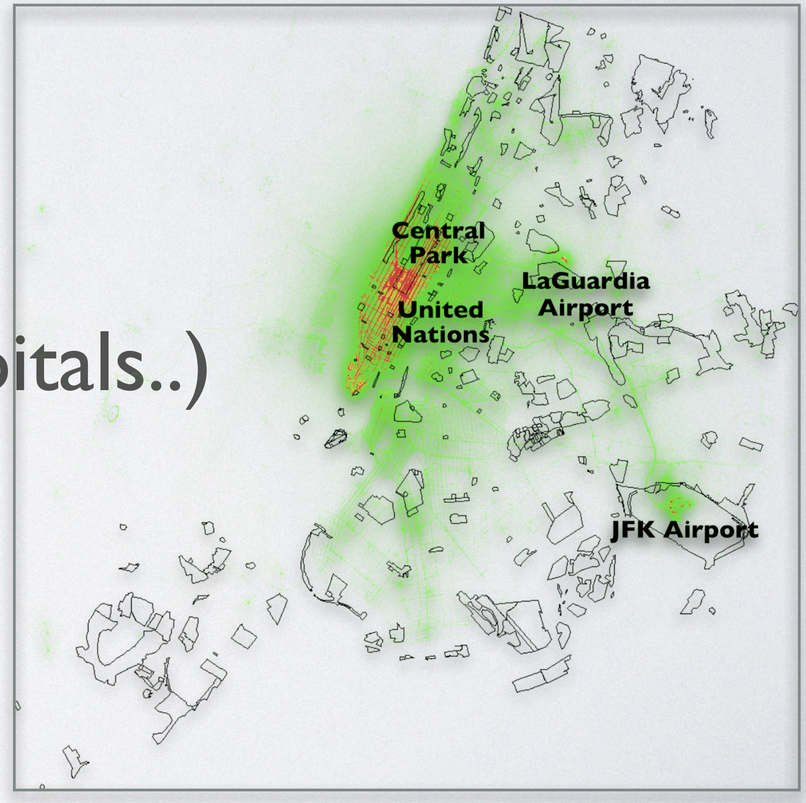
Spatial Visual Analytics

- Scalable visualization: visualize BILLION objects on Gigapixel map
- Customizable visualization: manipulate pixels at scale



Spatial Data Mining Example

- Spatial co-location pattern mining in Spark
  - Use spatial join to build a whole data mining application
  - Use map algebra to visualize the result
  - Data: Taxi pick ups (1 billion)
  - Data: NYC landmarks (300, airports, hospitals..)



https://github.com/jiayuasu/GeoSparkTemplateProject/tree/master/geospark-analysis

Arizona State University

Outline

Instruction

Development

Step-by-Step Tutorial

- Amazon Web Services EC2 GeoSpark cluster
- Apache Zeppelin interactive notebook
- Try GeoSpark code in your browser
- Run the job in the cloud

GeoSpark

Arizona State University

Zeppelin Notebook Job Search anonymous

GeoSpark_Tutorial/1_SQL

Head default

Took 0 sec. Last updated by anonymous at November 03 2019, 11:32:11 PM.

```
%sql
-- Coordinate Reference System transformation
-- From degree-based CRS to meter-based CRS, Chicago (long,lat)
CREATE OR REPLACE TEMP VIEW county_meter AS
SELECT ST_Transform(shape, 'epsg:4326','epsg:3857') AS shape, county_name,
    ST_Transform(ST_Point(-87.637437, 41.888812), 'epsg:4326','epsg:3857') as
    chicago
FROM county_wkt
```

FINISHED

Took 1 sec. Last updated by anonymous at November 03 2019, 11:32:12 PM.

```
%sql
-- Coordinate Reference System transformation
-- Compute the distance between each county and Chicago
CREATE OR REPLACE TEMP VIEW county_wkt AS
SELECT shape, county_name, ST_Distance(ST_Transform(shape, 'epsg:4326','epsg
    :3857'), ST_Transform(ST_Point(-87.637437, 41.888812), 'epsg:4326','epsg
    :3857')) AS distance_chicago
FROM county_wkt
```

FINISHED

Took 0 sec. Last updated by anonymous at November 03 2019, 11:32:14 PM.

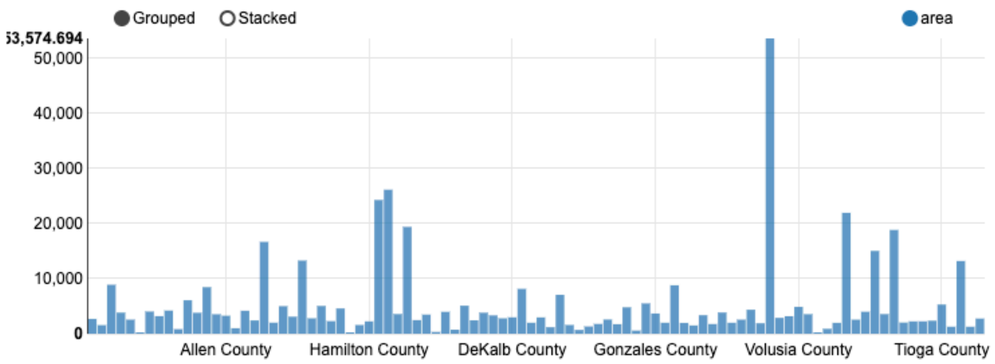
```
%sql
-- Coordinate Reference System transformation
-- Find the county which is within 500 kilometers of Chicago.
-- The unit of 500 KM is meter.
SELECT *
FROM county_wkt
WHERE distance_chicago < 500000
```

SPARK JOBS FINISHED

```
%sql
-- Compute the area of a county. Unit: Square KM
SELECT county_name, ST_Area(shape)/1000000 AS area
FROM county_meter
```

SPARK JOBS FINISHED

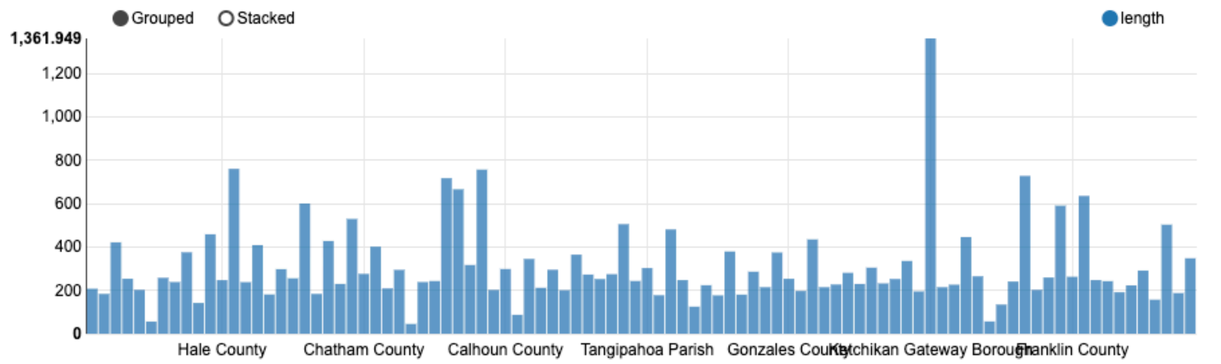
settings




```
%sql
-- Compute the perimeter of a county. Unit: KM
SELECT county_name, ST_Length(shape)/1000 AS length
FROM county_meter
```

SPARK JOBS FINISHED

settings



Arizona State University

Zeppelin Notebook Job Search anonymous

GeoSpark_Tutorial/2_RDD

Head default

```
// GeoSpark provides an Adapter class
val pointdf = Adapter.toDf(pointrdd, spark)
pointdf.createOrReplaceTempView("area_landmark")
pointdf.show
```

```
+--------------------+
|            geometry|
+--------------------+
|POINT (-88.331492...|
|POINT (-88.175933...|
|POINT (-88.388954...|
|POINT (-88.221102...|
|POINT (-88.323995...|
|POINT (-88.231077...|
|POINT (-88.349276...|
|POINT (-88.304259...|
|POINT (-88.182481...|
|POINT (-86.955186...|
|POINT (-87.059169...|
|POINT (-87.251219...|
|POINT (-87.534883...|
|POINT (-87.49702 ...|
|POINT (-88.280149...|
```

Took 1 sec. Last updated by anonymous at November 03 2019, 6:36:39 PM.

```
// GeoSpark provides an Adapter class
val polygondf_wkb = Adapter.toDf(polygonrdd_wkb, spark)
polygondf_wkb.createOrReplaceTempView("county_wkb")
polygondf_wkb.show
```

```
+--------------------+---+---+--------+-----+--------+--------------------+---+
---+-----+----+-----+----+----+---------+-------+----------+-----------+
|            geometry|_c1|_c2|     _c3|  _c4|     _c5|                 _c6|_c7|
_c8|  _c9|_c10| _c11|_c12|_c13|      _c14|    _c15|       _c16|       _c17|
+--------------------+---+---+--------+-----+--------+--------------------+---+
---+-----+----+-----+----+----+---------+-------+----------+-----------+
|POLYGON ((-97.019...| 31|039|00835841|31039|     Cuming|       Cuming County| 06|
H1|G4020|    |     |    |   A|1477895811|10447360|+41.9158651|-096.7885168|
|POLYGON ((-123.43...| 53|069|01513275|53069|  Wahkiakum|    Wahkiakum County| 06|
H1|G4020|    |     |    |   A| 68213887|1|61658258|+46.2946377|-123.4244583|
|POLYGON ((-104.56...| 35|011|00933054|35011|    De Baca|      De Baca County| 06|
H1|G4020|    |     |    |   A|6015539696|29159492|+34.3592729|-104.3686961|
|POLYGON ((-96.910...| 31|109|00835876|31109|  Lancaster|    Lancaster County| 06|
H1|G4020| 339|30700|    |   A|2169240202|22877180|+40.7835474|-096.6886584|
|POLYGON ((-98.273...| 31|129|00835886|31129|   Nuckolls|     Nuckolls County| 06|
H1|G4020|    |     |    |   A|1489645187| 1718484|+40.1764918|-098.0468422|
|POLYGON ((-65.910...| 72|085|01804523|72085|Las Piedras|Las Piedras Munic...| 13|
H1|G4020| 490|41980|    |   A| 87748363|   32509|+18.1871483|-065.8711890|
```

Took 1 sec. Last updated by anonymous at November 03 2019, 6:36:40 PM.

```
// GeoSpark provides an Adapter class
val polygondf_wkt = Adapter.toDf(polygonrdd_wkt, spark)
polygondf_wkt.createOrReplaceTempView("county_wkt")
polygondf_wkt.show
```

```
+--------------------+---+---+--------+-----+--------+--------------------+---+
---+-----+----+-----+----+----+---------+-------+----------+-----------+
|            geometry|_c1|_c2|     _c3|  _c4|     _c5|                 _c6|_c7|
_c8|  _c9|_c10| _c11|_c12|_c13|      _c14|    _c15|       _c16|       _c17|
+--------------------+---+---+--------+-----+--------+--------------------+---+
---+-----+----+-----+----+----+---------+-------+----------+-----------+
|POLYGON ((-97.019...| 31|039|00835841|31039|     Cuming|       Cuming County| 06|
H1|G4020|    |     |    |   A|1477895811|10447360|+41.9158651|-096.7885168|
|POLYGON ((-123.43...| 53|069|01513275|53069|  Wahkiakum|    Wahkiakum County| 06|
H1|G4020|    |     |    |   A| 68213887|1|61658258|+46.2946377|-123.4244583|
|POLYGON ((-104.56...| 35|011|00933054|35011|    De Baca|      De Baca County| 06|
H1|G4020|    |     |    |   A|6015539696|29159492|+34.3592729|-104.3686961|
|POLYGON ((-96.910...| 31|109|00835876|31109|  Lancaster|    Lancaster County| 06|
H1|G4020| 339|30700|    |   A|2169240202|22877180|+40.7835474|-096.6886584|
|POLYGON ((-98.273...| 31|129|00835886|31129|   Nuckolls|     Nuckolls County| 06|
H1|G4020|    |     |    |   A|1489645187| 1718484|+40.1764918|-098.0468422|
|POLYGON ((-65.910...| 72|085|01804523|72085|Las Piedras|Las Piedras Munic...| 13|
H1|G4020| 490|41980|    |   A| 87748363|   32509|+18.1871483|-065.8711890|
```

Took 1 sec. Last updated by anonymous at November 03 2019, 6:36:41 PM.

```
// Build a spatial index
// Build a quad-tree index for pointrdd
pointrdd.buildIndex(IndexType.RTREE, false)
println(pointrdd.indexedRawRDD.count())
```

SPARK JOBS FINISHED

```
2
```

Took 1 sec. Last updated by anonymous at November 03 2019, 6:36:42 PM.

```
// Build a spatial index
// Build a r-tree index for polygonrdd_wkb
polygonrdd_wkb.buildIndex(IndexType.QUADTREE, false)
println(polygonrdd_wkb.indexedRawRDD.count())
```

SPARK JOBS FINISHED

```
2
```

Took 0 sec. Last updated by anonymous at November 03 2019, 6:36:42 PM.

```
// Build a spatial index
// Build a quad-tree index for polygonrdd_wkb
polygonrdd_wkt.buildIndex(IndexType.RTREE, false)
println(polygonrdd_wkt.indexedRawRDD.count())
```

SPARK JOBS FINISHED

```
2
```

Took 1 sec. Last updated by anonymous at November 03 2019, 6:36:43 PM.

```
// Spatial range query
// Find landmarks within Birmingham City
val queryWindow = new Envelope(-87.111257, -86.554093, 33.339587, 33.645426)
val queryResultRdd = RangeQuery.SpatialRangeQuery(pointrdd, queryWindow, false, true)
println(queryResultRdd.count())

// We can check Spark web ui to see the detailed run time
```

SPARK JOBS FINISHED

```
// Spatial range query
// Cache the built index and use it many times
val queryWindow = new Envelope(-87.111257, -86.554093, 33.339587, 33.645426)

pointrdd.indexedRawRDD.cache()

val queryResultRdd_2 = RangeQuery.SpatialRangeQuery(pointrdd, queryWindow, false, true)
println(queryResultRdd_2.count())
```

SPARK JOBS FINISHED

Arizona State University

Zeppelin Notebook Job Search anonymous

GeoSpark_Tutorial/3_Join

Head default

```
var polygondf_wkt = spark.read.format("csv").option("delimiter", "\t").option("header", "false").load("/home/ubuntu/data/county_small.tsv")
polygondf_wkt.createOrReplaceTempView("county_wkt")
polygondf_wkt.show()
```

```
%sql
-- area_landmark constructor
CREATE OR REPLACE TEMP VIEW area_landmark AS
SELECT ST_Point(lon, lat) as shape, 'I am a point'
FROM area_landmark
```

FINISHED

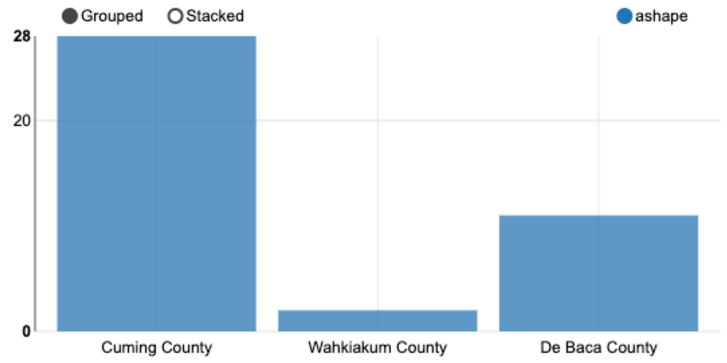
```
%sql
-- county constructor
CREATE OR REPLACE TEMP VIEW county_wkt AS
SELECT ST_GeomFromWKT(_c0) as shape, _c6 AS county_name
FROM county_wkt
```

FINISHED

```
%sql
-- Run a spatial join query using ST_Contains
-- Compute the number of landmarks per county
-- Visualize the results
SELECT c.shape AS cshape, c.county_name,  a.shape AS ashape
FROM area_landmark a, county_wkt c
WHERE ST_Contains(c.shape, a.shape)
LIMIT 100
-- Zeppelin cannot handle too many viz results, we have to LIMIT the output size
```

SPARK JOBS FINISHED

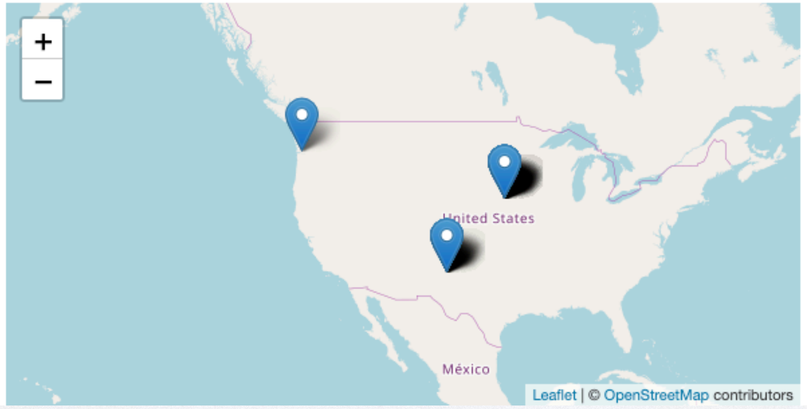
settings




```
%sql
-- Run a spatial join query using ST_Contains
-- Compute the number of landmarks per county
-- Visualize the results
SELECT c.shape AS cshape, c.county_name,  a.shape AS ashape
FROM area_landmark a, county_wkt c
WHERE ST_Contains(c.shape, a.shape)
LIMIT 100
-- Zeppelin cannot handle too many viz results, we have to LIMIT the output size
```

SPARK JOBS FINISHED

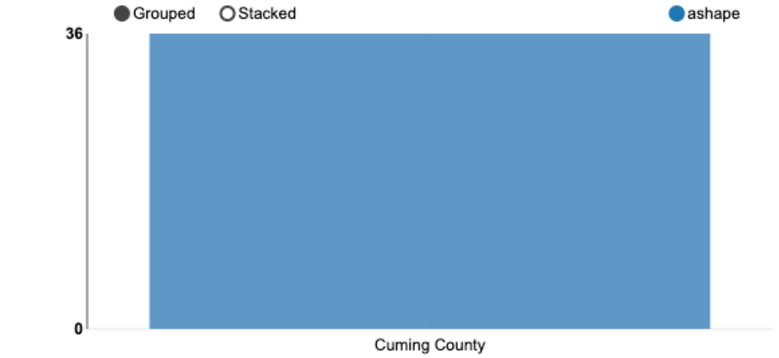
settings




```
%sql
-- Run a distance join query using ST_Distance
-- Area landmarks with 100 km of each county
-- This requires CRS transformation which is omitted here
SELECT c.shape AS cshape, c.county_name,  a.shape AS ashape
FROM area_landmark a, county_wkt c
WHERE ST_Distance(c.shape, a.shape) < 1000000
LIMIT 100
```

SPARK JOBS FINISHED

settings



Arizona State University

Zeppelin Notebook Job Search anonymous

GeoSpark_Tutorial/4_GeoViz

Head default

```
-- Render colors
CREATE OR REPLACE TEMP VIEW colors AS
------ Heat map
SELECT pixel, ST_Colorize(weight, (SELECT max(weight) FROM pixelaggregates)) AS color
------ Scatter plot
-- SELECT pixel, ST_Colorize(1000, (SELECT max(weight) FROM pixelaggregates)) AS color
FROM pixelaggregates
```

```
-- Generate images
CREATE OR REPLACE TEMP VIEW images AS
SELECT ST_Render(pixel, color) AS image
FROM colors
```

```
%sql
-- Encode the image to a format that Zeppelin understands
SELECT ST_EncodeImage(image) AS image,(SELECT ST_AsText(ST_PolygonFromEnvelope(-126.790180,24.863836,-64.630926,50.000))) AS boundary
FROM images
```

SPARK JOBS FINISHED

settings

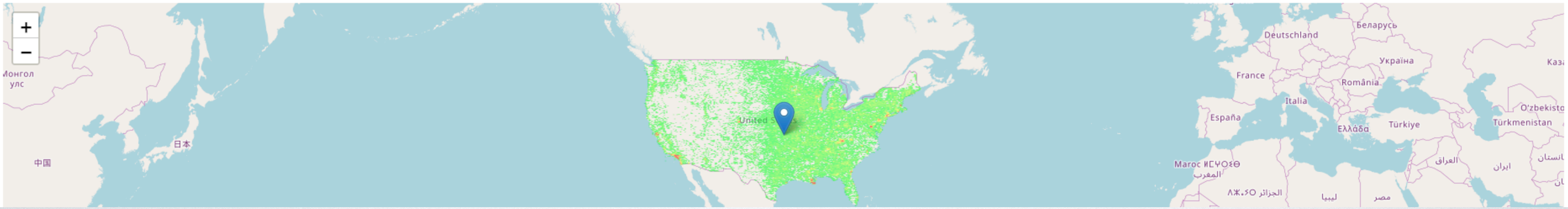
Available Fields

image boundary

mapimage: image ×

geometry: boundary ×

info



Arizona State University

Zeppelin Notebook Job Search anonymous

GeoSpark_Tutorial/5_CoLocation_Pattern_Mining

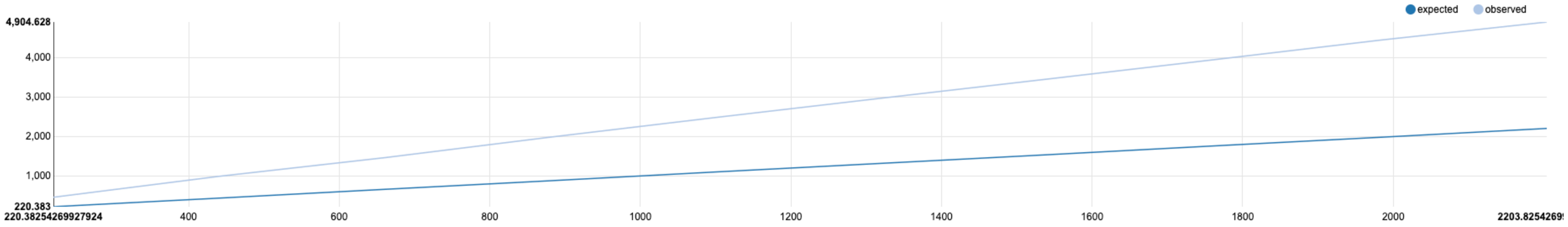
Head default

```
  var adjacentMatrix = JoinQuery.DistanceJoinQueryFlat(tripRDD, bufferedArealmRDD,true,true)
  //      Uncomment the following two lines if you want to see what the join result looks like in SparkSQL
//   var adjacentMatrixDf = Adapter.toDf(adjacentMatrix, spark)
//   adjacentMatrixDf.show()
  var observedK = adjacentMatrix.count()*area*1.0/(arealmRDD.approximateTotalCount*tripRDD.approximateTotalCount)
  var observedL = Math.sqrt(observedK/Math.PI)
  var expectedL = currentDistance
  var colocationDifference = observedL  - expectedL
  var colocationStatus = {if (colocationDifference>0) "Co-located" else "Dispersed"}
  result = result :+ (currentDistance, observedL, expectedL, colocationStatus)
}
import spark.implicits._
var df = result.toDF("distance(meter)", "observed", "expected", "coLocationStatus")
// df.show
df.createOrReplaceTempView("ripleyk")
```

FINISHED

```
%sql
-- Plot the line curve the result
-- Observed K is higher than the expected K
-- Colocation pattern exists in these distance points
SELECT *
FROM ripleyk
```

settings



Tutorial Schedule

- 13:15-14:00 : Session 3 : Tutorial 3 (Instruction)
- 14:00-14:45 : Session 4 : Tutorial 3 (Development)