Crunching and visualizing Big Data

on a Computer Cluster

Joana Simões

April 10, 2015

https://github.com/doublebyte1/workshop_bdigital

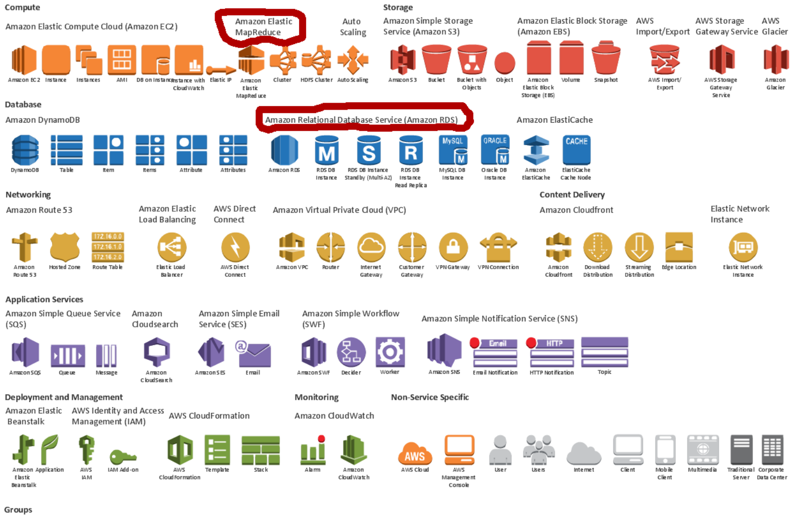
Table of Contents

1. Introduction
2. Importing a Spatial-Temporal Series
3. Recovering the Spatial Attributes
4. Putting it All Together
5. Piping the Results into the Outside World

Motivation

- Problem: the increasing volume of information, by explosion of traditional sources + new sources
- Target: fast query responses, which require a scalable architecture
- Possible solution: support clusters on a cost-effective architecture, such as commodity clusters or cloud environments

Cloud Services



A thought...

First the use case, then the tools.

Use Case

- Study spatial and temporal patterns of road traffic accidents.
- Relate target variable (accident) with context variables (e.g.: weather, proximity to SPI).

Use Case

- Study spatial and temporal patterns of road traffic accidents.
- Relate target variable (accident) with context variables (e.g.: weather, proximity to SPI).
- In most (Big) Data Analysis 80% of the effort is devoted to the Extract-Transform-Load (ETL) process
- ETL: process responsible for pulling data out of the source systems and placing it into a data warehouse

Use Case

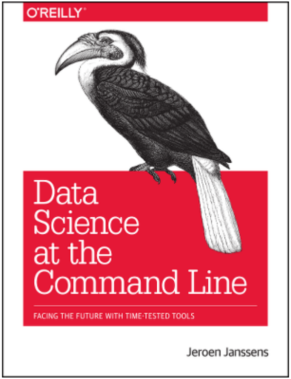
- Study spatial and temporal patterns of road traffic accidents.
- Relate target variable (accident) with context variables (e.g.: weather, proximity to SPI).
- In most (Big) Data Analysis 80% of the effort is devoted to the Extract-Transform-Load (ETL) process
- ETL: process responsible for pulling data out of the source systems and placing it into a data warehouse
  - **Extract** data from different source systems and convert it into one consolidated data warehouse format which is ready for transformation processing
  - **Transform**: cleaning, filtering, splitting a column, joining data, apply validation, apply rules, etc
  - **Load**: into the data warehouse, repository or reporting applications

Another thought...

There are no free lunches.

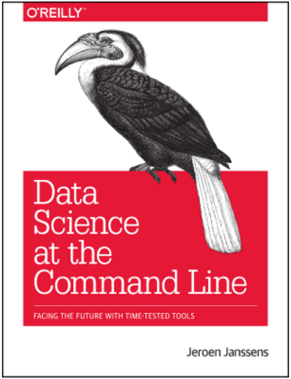
Scalable ML Platforms

- ML platforms may provide high-level data import tools:


O'REILLY
Data Science at the Command Line
FACING THE FUTURE WITH TIME-TESTED TOOLS
Jeroen Janssens

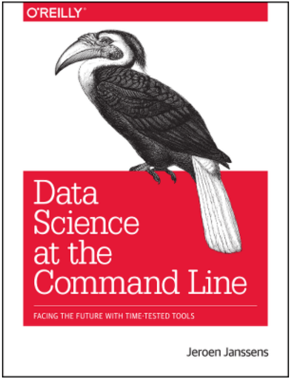
Scalable ML Platforms

- ML platforms may provide high-level data import tools:
    - They generally trade ease of use for flexibility


O'REILLY
Data Science at the Command Line
FACING THE FUTURE WITH TIME-TESTED TOOLS
Jeroen Janssens

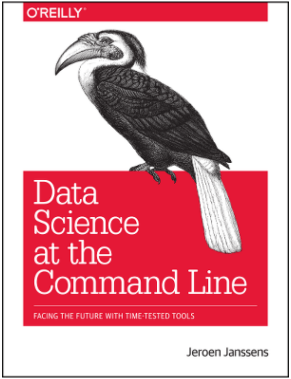
Scalable ML Platforms

- ML platforms may provide high-level data import tools:
  - They generally trade ease of use for flexibility
  - They may have a cost ($$$)

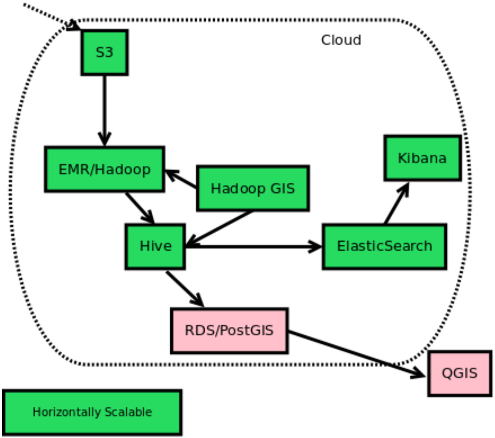

O'REILLY
Data Science at the Command Line
FACING THE FUTURE WITH TIME-TESTED TOOLS
Jeroen Janssens

Scalable ML Platforms

- ML platforms may provide high-level data import tools:
  - They generally trade ease of use for flexibility
  - They may have a cost ($$$)
- To ensure maximum flexibility, we should be able to link together many tools, often using the command line.

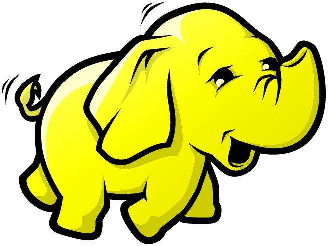

O'REILLY
Data Science at the Command Line
FACING THE FUTURE WITH TIME-TESTED TOOLS
Jeroen Janssens

Stack


Cloud
S3
EMR/Hadoop
Hadoop GIS
Kibana
Hive
ElasticSearch
RDS/PostGIS
QGIS
Horizontally Scalable

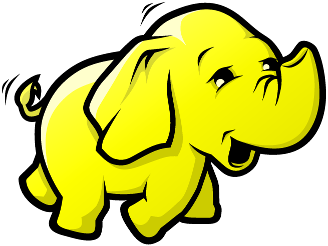
Apache Hadoop

- framework for distributed storage and processing of (Big) Data on computer Clusters



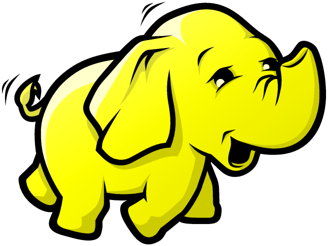
Apache Hadoop

- framework for distributed storage and processing of (Big) Data on computer Clusters
  - **Storage:** HDFS
  - **Processing:** MapReduce



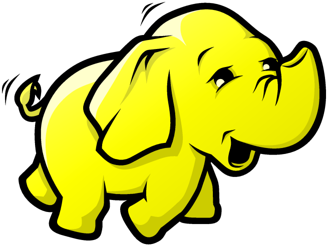
Apache Hadoop

- framework for distributed storage and processing of (Big) Data on computer Clusters
  - **Storage:** HDFS
  - **Processing:** MapReduce
- It features a FOS license (Apache 2.0)



Apache Hadoop

- framework for distributed storage and processing of (Big) Data on computer Clusters
  - **Storage:** HDFS
  - **Processing:** MapReduce
- It features a FOS license (Apache 2.0)
- EMR is an Amazon service that uses Hadoop



Apache Hive

- Data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis.

Apache Hive

- Data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis.

Apache Hive

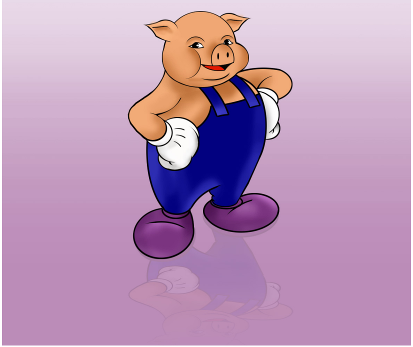
- Data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis.
- HiveQL: query language based on SQL
  - An internal compiler translates HiveQL statements into MapReduce jobs
- It features a FOS license (Apache 2.0)

Apache Hive

- Data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis.
- HiveQL: query language based on SQL
    - An internal compiler translates HiveQL statements into MapReduce jobs
- It features a FOS license (Apache 2.0)
- EMR features an Hive installation

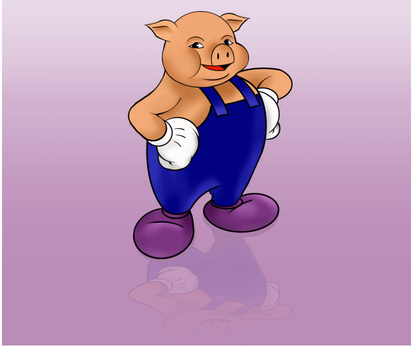
Apache Pig

- High-level platform for creating MapReduce jobs over Hadoop.



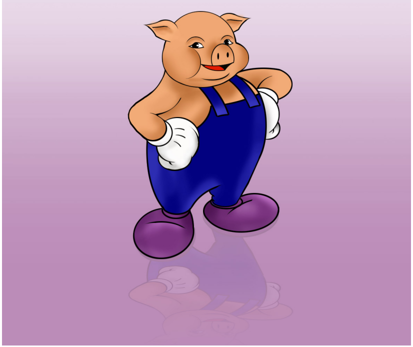
Apache Pig

- High-level platform for creating MapReduce jobs over Hadoop.
- It features an interactive mode and a batch mode



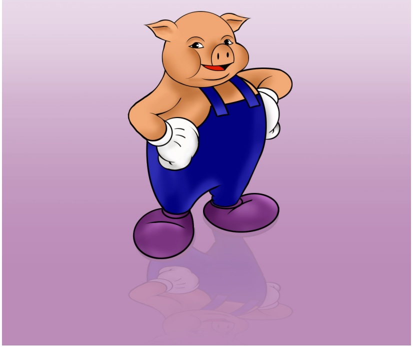
Apache Pig

- High-level platform for creating MapReduce jobs over Hadoop.
- It features an interactive mode and a batch mode
- It uses lazy evaluation



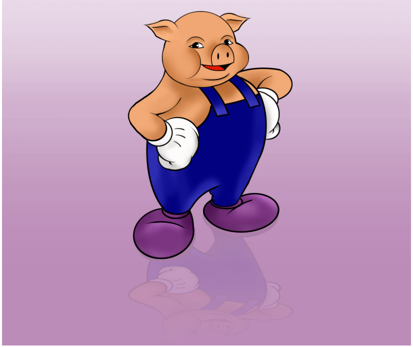
Apache Pig

- High-level platform for creating MapReduce jobs over Hadoop.
- It features an interactive mode and a batch mode
- It uses lazy evaluation
- Pig Latin is procedural



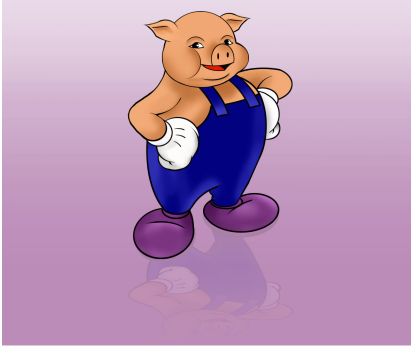
Apache Pig

- High-level platform for creating MapReduce jobs over Hadoop.
- It features an interactive mode and a batch mode
- It uses lazy evaluation
- Pig Latin is procedural
- It features a FOS license (Apache 2.0)



Apache Pig

- High-level platform for creating MapReduce jobs over Hadoop.
- It features an interactive mode and a batch mode
- It uses lazy evaluation
- Pig Latin is procedural
- It features a FOS license (Apache 2.0)
- EMR may install Pig



Spatial Support

Hadoop GIS

- FOS toolkit for ”Big Spatial Data Analytics”.

Hadoop GIS

- FOS toolkit for ”Big Spatial Data Analytics”.
- Hosted on github.

Hadoop GIS

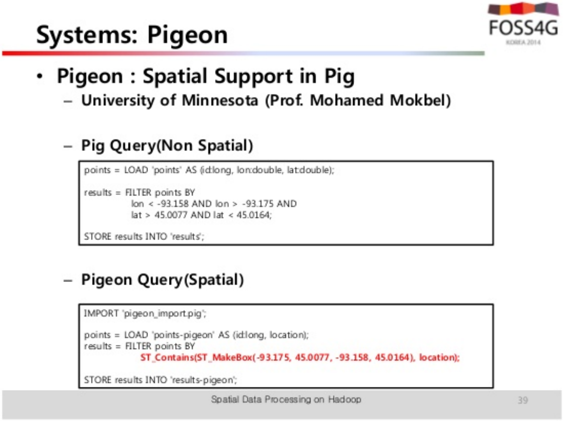
- FOS toolkit for ”Big Spatial Data Analytics”.
- Hosted on github.
- It consists in three libraries:

Hadoop GIS

- FOS toolkit for ”Big Spatial Data Analytics”.
- Hosted on github.
- It consists in three libraries:
    - Esri Geometry API for Java.
    - Geoprocessing Tools for Hadoop.
    - **Spatial Framework for Hadoop (SFH):** extends Hive to to enable spatial queries and geometry types.

Pigeon

- Wrapper around the geometry API from ESRI.
- Adds spatial support to Pig Latin.



Systems: Pigeon

FOSS4G KOREA 2014

- Pigeon : Spatial Support in Pig
  - University of Minnesota (Prof. Mohamed Mokbel)
  - Pig Query(Non Spatial)

```
points = LOAD 'points' AS (id:long, lon:double, lat:double);

results = FILTER points BY
          lon < -93.158 AND lon > -93.175 AND
          lat > 45.0077 AND lat < 45.0164;

STORE results INTO 'results';
```

  - Pigeon Query(Spatial)

```
IMPORT 'pigeon_import.pig';

points = LOAD 'points-pigeon' AS (id:long, location);
results = FILTER points BY
            ST_Contains(ST_MakeBox(-93.175, 45.0077, -93.158, 45.0164), location);

STORE results INTO 'results-pigeon';
```

Spatial Data Processing on Hadoop 39

PostGIS

- Spatial extension for the RDBMS PostgreSQL
- Free and Open Source License (GPL 2.0)

PostGIS

- Spatial extension for the RDBMS PostgreSQL
- Free and Open Source License (GPL 2.0)
- Standards support

PostGIS

- Spatial extension for the RDBMS PostgreSQL
- Free and Open Source License (GPL 2.0)
- Standards support
- High Performance

PostGIS

- Spatial extension for the RDBMS PostgreSQL
- Free and Open Source License (GPL 2.0)
- Standards support
- High Performance
- Famous users: Foursquare, Instagram, CartoDB.

E(L)K

- **ElasticSearch**: Search engine based on Lucene (RESTful, JSON).

E(L)K

- **ElasticSearch**: Search engine based on Lucene (RESTful, JSON).
- **Logstash**: tool for managing events and logs.

E(L)K

- **ElasticSearch**: Search engine based on Lucene (RESTful, JSON).
- **Logstash**: tool for managing events and logs.
- **Kibana**: data visualization platform (custom dashboards).

Practical

Hands-on

Connect to the Cluster

- Micro-task: Connect to the cluster using SSH

Connect to the Cluster

- Micro-task: Connect to the cluster using SSH
  - Start the SSH client (differences may apply), and connect using the pem certificate.
  - `https://drive.google.com/a/bdigital.org/file/d/0B5kkho5DSzlyanBkUzgtRWh20VE/view?usp=sharing`

From S3 to HDFS

- Micro-task: Understand the dataset structure
    - A sample dataset is stored on an S3 bucket: `s3n://workshop-bdsd/accidents/`

From S3 to HDFS

- Micro-task: Understand the dataset structure
  - A sample dataset is stored on an S3 bucket: `s3n://workshop-bdsd/accidents/`
  - **`https://s3-eu-west-1.amazonaws.com/workshop-bdsd/accidents/accidents_sample.csv`**

From S3 to HDFS

- Micro-task: Understand the dataset structure
  - A sample dataset is stored on an S3 bucket: `s3n://workshop-bdsd/accidents/`
  - **`https://s3-eu-west-1.amazonaws.com/workshop-bdsd/accidents/accidents_sample.csv`**
  - Download and view dataset

From S3 to HDFS (cont.)

- Micro-task: Create a table linking to the data
  - Enter hive and create an external table linking to the S3 bucket

From S3 to HDFS (cont.)

- Micro-task: Create a table linking to the data
  - Enter hive and create an external table linking to the S3 bucket
  - Use the CSV serde to parse the table structure

From S3 to HDFS (cont.)

- Micro-task: Create a table linking to the data
    - Enter hive and create an external table linking to the S3 bucket
    - Use the CSV serde to parse the table structure
        - separator char
        - quote char
        - headers

From S3 to HDFS (cont.)

- Micro-task: Create a table linking to the data
  - Enter hive and create an external table linking to the S3 bucket
  - Use the CSV serde to parse the table structure
    - separator char
    - quote char
    - headers
  - View imported data

From S3 to HDFS (cont.)

- Micro-task: Type Mapping

From S3 to HDFS (cont.)

- Micro-task: Type Mapping
    - Create an empty table with correct types

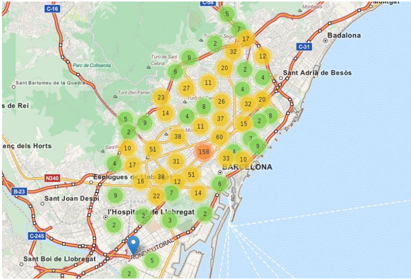
From S3 to HDFS (cont.)

- Micro-task: Type Mapping
  - Create an empty table with correct types
  - Insert data from accidents_import

From S3 to HDFS (cont.)

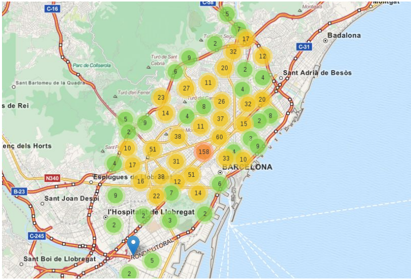
- Micro-task: Type Mapping
  - Create an empty table with correct types
  - Insert data from accidents_import
  - View table

What is so "Special" about Spatial



What is so "Special" about Spatial

- Location attributes allow us to detect spatial patterns
- Location also works as a "key", allowing us to connect with other datasets



Analysis of the Spatial Attributes

- Spatial Attributes are encoded as coordinates in ”d_coord_geo_impacte”

Analysis of the Spatial Attributes

- Spatial Attributes are encoded as coordinates in ”d_coord_geo_impacte”
- Problems with this field:

Analysis of the Spatial Attributes

- Spatial Attributes are encoded as coordinates in ”d_coord_geo_impacte”
- Problems with this field:
  - Unstructured: needs parsing
  - Inconsistent format (order of coordinates, separator)
  - Invalid values (e.g.: 77, names)
  - No metadata

Analysis of the Spatial Attributes

- Spatial Attributes are encoded as coordinates in ”d_coord_geo_impacte”
- Problems with this field:
  - Unstructured: needs parsing
  - Inconsistent format (order of coordinates, separator)
  - Invalid values (e.g.: 77, names)
  - No metadata
  - Mixed CRS:

Analysis of the Spatial Attributes

- Spatial Attributes are encoded as coordinates in ”d_coord_geo_impacte”
- Problems with this field:
    - Unstructured: needs parsing
    - Inconsistent format (order of coordinates, separator)
    - Invalid values (e.g.: 77, names)
    - No metadata
    - Mixed CRS:
        - WGS84 (EPSG:4326)
        - European Grid (EPSG:5554)
        - European Grid encoded by the police using an *ad-hoc* format

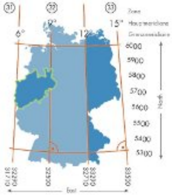
Analysis of the Spatial Attributes

- Spatial Attributes are encoded as coordinates in "d_coord_geo_impacte"
- Problems with this field:
  - Unstructured: needs parsing
  - Inconsistent format (order of coordinates, separator)
  - Invalid values (e.g.: 77, names)
  - No metadata
  - Mixed CRS:
    - WGS84 (EPSG:4326)
    - European Grid (EPSG:5554)
    - European Grid encoded by the police using an *ad-hoc* format
    - $lon = y/1000 + 400000$
    - $lat = y/1000 + 4500000$

CRS

World Geodetic System (WGS84, EPSG:4326): standard for use in cartography, geodesy, and navigation; reference CRS for GPS.

European Terrestrial Reference System 1989 (ETRS89, EPSG:5554): proposed, multipurpose Pan-European mapping standard; based on the ETRS89 Lambert Azimuthal Equal-Area projection coordinate reference system



Objective

- Separate lat, long fields and map them to correct types
- Remove invalid values
- Convert all coordinates into a single CRS (WGS84)

Exporting Data to Pig

- Pig uses filters to subset the data
- To merge back the subsetted data, we can use joins by a common field
- Micro-task: Export the data

Exporting Data to Pig

- Pig uses filters to subset the data
- To merge back the subsetted data, we can use joins by a common field
- Micro-task: Export the data
  - Create a copy of the accidents table, with an id field (joins).

Exporting Data to Pig

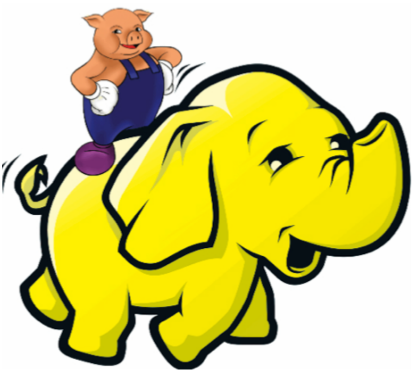
- Pig uses filters to subset the data
- To merge back the subsetted data, we can use joins by a common field
- Micro-task: Export the data
  - Create a copy of the accidents table, with an id field (joins).
  - Export this table into a tsv

Exporting Data to Pig

- Pig uses filters to subset the data
- To merge back the subsetted data, we can use joins by a common field
- Micro-task: Export the data
    - Create a copy of the accidents table, with an id field (joins).
    - Export this table into a tsv
    - Store it in HDFS (if needed)
    - View exported data

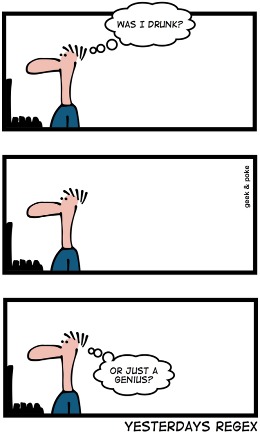
Presenting the Pig Script

- Subsets the coordinate list, using filters
- Detects each coordinate "type", using regular expressions
- In the case of grid encoded, it applies a formula to decode back into grid
- Stores the results into separate files, in HDFS



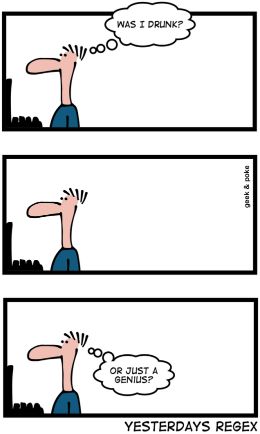
REGEX

- Sequence of characters that forms a search pattern, mainly for use in pattern matching with strings, or string matching



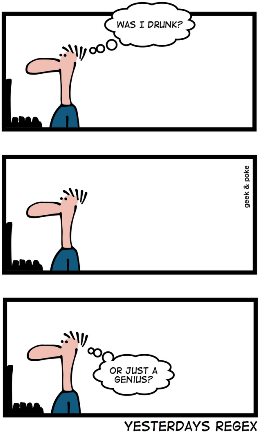
REGEX

- Sequence of characters that forms a search pattern, mainly for use in pattern matching with strings, or string matching
- REGEX_EXTRACT(D_COORD_C
  z]',0)



REGEX

- Sequence of characters that forms a search pattern, mainly for use in pattern matching with strings, or string matching
- REGEX_EXTRACT(D_COORD_(z]',0)
- '[A-z]'



Running Pig

- Micro-task: run pig script

Running Pig

- Micro-task: run pig script
  - Download script from S3: `https://s3-eu-west-1.amazonaws.com/workshop-bdsd/recover_geography.pig`

Running Pig

- Micro-task: run pig script
  - Download script from S3: `https://s3-eu-west-1.amazonaws.com/workshop-bdsd/recover_geography.pig`
  - Edit script and **ammend paths**

Running Pig

- Micro-task: run pig script
  - Download script from S3: `https://s3-eu-west-1.amazonaws.com/workshop-bdsd/recover_geography.pig`
  - Edit script and **ammend paths**
  - Run script

Running Pig

- Micro-task: run pig script
  - Download script from S3: `https://s3-eu-west-1.amazonaws.com/workshop-bdsd/recover_geography.pig`
  - Edit script and **ammend paths**
  - Run script
  - Check output files

Importing Data Back into Hive

- Micro-task: Create tables linking to pig output

Importing Data Back into Hive

- Micro-task: Create tables linking to pig output
  - Create table with wgs84 data
  - Create table with grid data
  - Create table with police-decoded data

Exporting data into PostGIS

- As of Hadoop GIS 2.0, CRS transformation is **not supported**
- We need to rely on another tool: PostGIS on RDS
- Micro-task: Export grid data to PostGIS

Exporting data into PostGIS

- As of Hadoop GIS 2.0, CRS transformation is **not supported**
- We need to rely on another tool: PostGIS on RDS
- Micro-task: Export grid data to PostGIS
    - Merge grid (grid + police) tables in a single table

Exporting data into PostGIS

- As of Hadoop GIS 2.0, CRS transformation is **not supported**
- We need to rely on another tool: PostGIS on RDS
- Micro-task: Export grid data to PostGIS
  - Merge grid (grid + police) tables in a single table
  - Exported merged table into TSV

Importing Data into PostGIS

- Micro-task: Import grid data into PostGIS

Importing Data into PostGIS

- Micro-task: Import grid data into PostGIS
    - Install the PSQL client
    - Log into RDS:

Importing Data into PostGIS

- Micro-task: Import grid data into PostGIS
  - Install the PSQL client
  - Log into RDS:
    - host: bdigitaldb.celqzuwfokoe.eu-west-1.rds.amazonaws.com
    - user: workshop
    - password: geohipster
    - database: workshop_bdigital

Importing Data into PostGIS

- Micro-task: Import grid data into PostGIS
  - Install the PSQL client
  - Log into RDS:
    - host: bdigitaldb.celqzuwfokoe.eu-west-1.rds.amazonaws.com
    - user: workshop
    - password: geohipster
    - database: workshop_bdigital
  - Create table to accomodate data

Importing Data into PostGIS

- Micro-task: Import grid data into PostGIS
  - Install the PSQL client
  - Log into RDS:
    - host: bdigitaldb.celqzuwfokoe.eu-west-1.rds.amazonaws.com
    - user: workshop
    - password: geohipster
    - database: workshop_bdigital
  - Create table to accomodate data
  - Copy data into table

CRS Transformation

- Micro-task: Convert all features in European grid to WGS84

CRS Transformation

- Micro-task: Convert all features in European grid to WGS84
  - Create geometry fields to accomodate geometry in the two CRS (Grid, WGS84)

CRS Transformation

- Micro-task: Convert all features in European grid to WGS84
  - Create geometry fields to accomodate geometry in the two CRS (Grid, WGS84)
    - Add columns
    - Set SRID
    - Create geometry index

CRS Transformation

- Micro-task: Convert all features in European grid to WGS84
  - Create geometry fields to accomodate geometry in the two CRS (Grid, WGS84)
    - Add columns
    - Set SRID
    - Create geometry index
  - Instantiate grid geometry

CRS Transformation

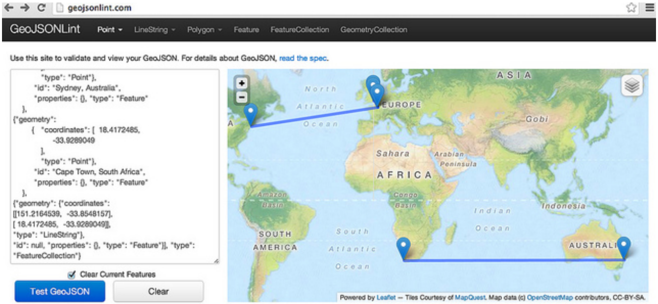
- Micro-task: Convert all features in European grid to WGS84
  - Create geometry fields to accomodate geometry in the two CRS (Grid, WGS84)
    - Add columns
    - Set SRID
    - Create geometry index
  - Instantiate grid geometry
  - Transform grid geometry into another CRS

CRS Transformation

- Micro-task: Convert all features in European grid to WGS84
  - Create geometry fields to accomodate geometry in the two CRS (Grid, WGS84)
    - Add columns
    - Set SRID
    - Create geometry index
  - Instantiate grid geometry
  - Transform grid geometry into another CRS
  - Export grid geometry in GeoJSON

GeoJSON

GeoJSON: is an open standard format for encoding collections of simple geographical features along with their non-spatial attributes using JavaScript Object Notation.



Use this site to validate and view your GeoJSON. For details about GeoJSON, read the spec.

```
        "type": "Point"},
      "id": "Sydney, Australia",
      "properties": {}, "type": "Feature"
  },
{"geometry":
     {   "coordinates": [  18.4172485,
              -33.9289049
        ],
        "type": "Point"},
      "id": "Cape Town, South Africa",
      "properties": {}, "type": "Feature"
  },
{"geometry": {"coordinates":
[[151.2164539,  -33.8548157],
[ 18.4172485,  -33.9289049]],
"type": "LineString"},
"id": null, "properties": {}, "type": "Feature"}], "type":
"FeatureCollection"}
```

Importing Data back into Hive

- Micro-task: Import transformed data

Importing Data back into Hive

- Micro-task: Import transformed data
  - Enter Hive

Importing Data back into Hive

- Micro-task: Import transformed data
  - Enter Hive
  - Create table linking to the PostGIS export

Importing Data back into Hive

- Micro-task: Import transformed data
  - Enter Hive
  - Create table linking to the PostGIS export
  - Create new table and instantiate geometry from GeoJSON

Joining Data

- Micro-task: Join imported coordinates with WGS84 coordinates and the rest of the dataset

Joining Data

- Micro-task: Join imported coordinates with WGS84 coordinates and the rest of the dataset
    - Join imported records with original table with all fields

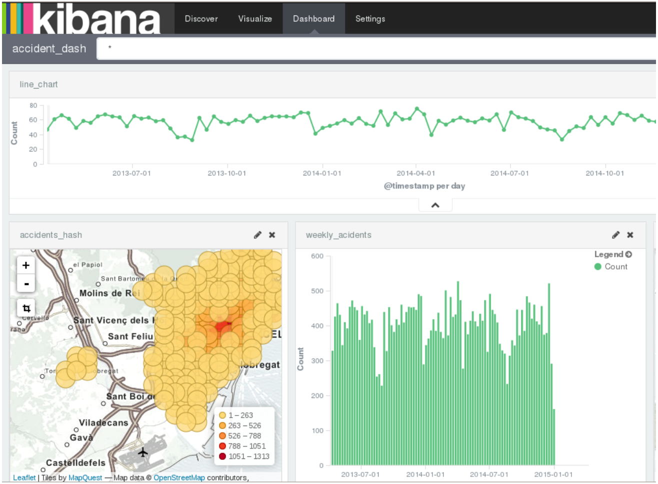
Joining Data

- Micro-task: Join imported coordinates with WGS84 coordinates and the rest of the dataset
    - Join imported records with original table with all fields
    - Merge imported records with WGS84 records, for a single table with unified geometry

ElasticSearch

Indexing the results in ElasticSearch

And now Kibana...!



Thank you for Listening!