CSI 4142: Introduction to Data Science

Project
Canadian Disaster Database

From
Alexandre Mizon 7680262
And
Daniel Audi 7382986

Presented to Professor Herna Viktor
April 5th 2018

Physical Design

We chose postgresSQL as our database management system. The script to create the tables for our data mart is included in Appendix A or our repository under the “Physical design” folder.

Data Staging

For data staging we decided to use python for programming language to perform any manipulation on the data. This include cleaning, normalizing, refactoring the data and adding it to 5 differents CSV corresponding to each of our data mart tables. We included below all the cleanup steps we followed as well as a high level schematic.

Data Cleanup Steps:

1. Removed empty rows and incomplete records not allowing us to try filling the missing information. Kept track of such records in the data cleanup registry located in the registry below.
2. Put 0 in all costs if estimated total cost and normalized cost were put to 0. Otherwise put unknown.
3. Complete missing EVENT END DATE column with either the information in the summary or by manual research.
4. For all non earthquake events replace 0 to blanks to not interfere with any possible data analysis, i.e. average magnitude.
5. For each location field, we used the powerful geoname library that makes an API call with the brut field retrieving all the necessary information such as city name, province/state, country, coordinates and populations.
   For the cases one cell contains more than one province and city we parse it based on some criterias, most of the time using the connector “and”, and make separate requests.

Data Cleanup Registry:

Row	Action	Description
1009	delete	Empty record
1010	delete	No information on the disaster or the location
1034	delete	Empty record
1035	delete	No information on the disaster or the location
1037	delete	Empty record
1038	delete	No information on the disaster or the location
1043	delete	Empty record
1044	delete	No information on the disaster or the location
1031	fill-up	Event summary was informative enough to make some research and complete most fields
274	update	Normalized location field by changing “across Canada’ to simply “Canada”
489	update	Normalized location field by changing “across Canada’ to simply “Canada”
525	update	Normalized location field by changing “across Canada’ to simply “Canada”
311	update	Normalized location field by changing “across Canada’ to simply “Canada”
488	update	Normalized location field by changing “across Canada’ to simply “Canada”
1008	complete	Event end date based on summary
967	update	Normalized location field by changing “across Canada’ to simply “Canada”
1033	complete	Event end date based on summary
1036	complete	Event end date based on summary
1037	complete	Event end date based on summary

High-Level One page schematic breakdown:

Identify starting and ending points

- CSV files ➡ PostgreSQL CanadianDisaster datamart

Label known data sources

- CanadianDisaster CSV file from Project
- Open source library (geoname) returning cities, population, provinces/states, and countries.

Include placeholders for sources yet to be determined

- Not applicable

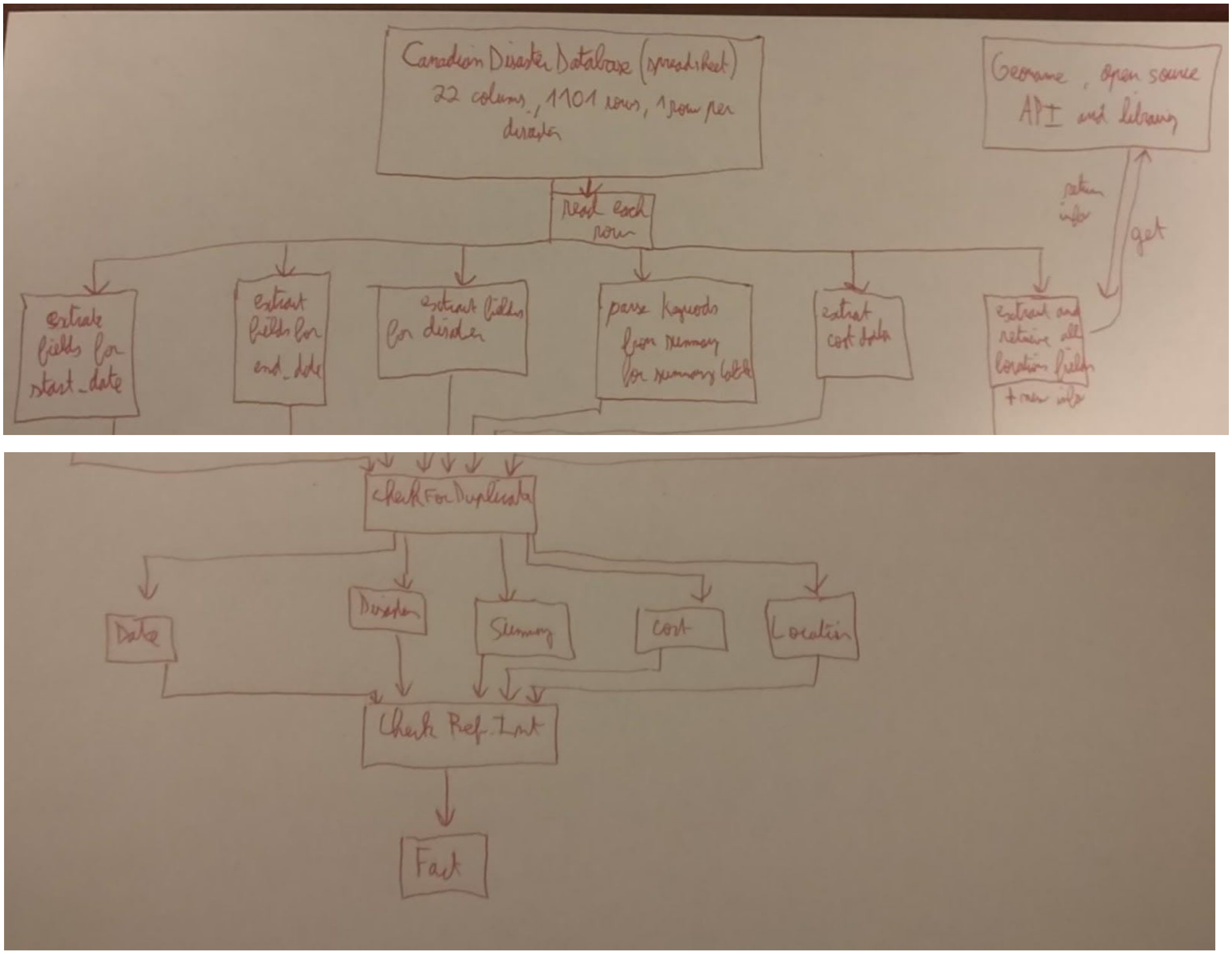
Label targets

- Facts, Location, Cost, Disaster, Date, Summary

Notes about known problems

- Extremely messy location data
- Provinces and city are mixed and don't always follow the same format
- Possibility of having more than one city and province in one record
- Possible use of ambiguous terms indicating a region rather than a city and province
- missing numerous event end dates

Flow:



OLAP Queries

Note: All our queries and results are available in the OLAP queries and results folders, in the zip file.

Drill down:

```
--DRILL DOWN
--YEAR DRILL DOWN:listing number of technology disasters each year, month day
select d.year,  DI.disaster_group, count(*) as total_count
from
```

```
fact f, date d, disaster DI
where f.start_date_key=D.date_key and f.disaster_key=DI.disaster_key and
DI.disaster_group='Technology'
group by (d.year, DI.disaster_group)
order by d.year DESC;

--SEASON DRILL DOWN:listing number of technology disasters each year, season
select d.year, d.season_canada, DI.disaster_group, count(*) as total_count
from
fact f, date d, disaster DI
where f.start_date_key=D.date_key and f.disaster_key=DI.disaster_key and
DI.disaster_group='Technology'
group by (d.year, d.season_canada, DI.disaster_group)
order by d.year, d.season_canada DESC;

--MONTH DRILL DOWN:listing number of technology disasters each year,season, month
select d.year, d.season_canada, d.month, DI.disaster_group, count(*) as total_count
from
fact f, date d, disaster DI
where f.start_date_key=D.date_key and f.disaster_key=DI.disaster_key and
DI.disaster_group='Technology'
group by (d.year, d.season_canada, d.month,DI.disaster_group)
order by d.year, d.season_canada, d.month DESC;

--DAY DRILL DOWN:listing number of technology disasters each year, season, month, day
select d.year, d.season_canada, d.month, d.day, DI.disaster_group, count(*) as total_count
from
fact f, date d, disaster DI
where f.start_date_key=D.date_key and f.disaster_key=DI.disaster_key and
DI.disaster_group='Technology'
group by (d.year, d.season_canada, d.month,d.day,DI.disaster_group)
order by d.year, d.season_canada, d.month , d.day DESC;
```

Roll-up:

```
--ROLL-UP

--ROLL UP BY CITY: Listing the total number of fatalities from 1990 to 2005 by city->
province -> country
select l.country, l.province, l.city, sum(f.fatalities) as total_fatalities
from fact f, location l, date d
where
f.location_key=l.location_key and f.start_date_key=d.date_key and d.year between
1990 and 2005
GROUP BY l.country, l.province, l.city
ORDER BY l.country, l.province, l.city ASC;

--ROLL UP BY PROVINCE: Listing the total number of fatalities from 1990 to 2005 by
province -> country
select l.country, l.province,  sum(f.fatalities) as total_fatalities
from fact f, location l, date d
where
f.location_key=l.location_key and f.start_date_key=d.date_key and d.year between
1990 and 2005
GROUP BY l.country, l.province
ORDER BY l.country, l.province ASC;

--ROLL UP BY YEAR: Listing the total number of fatalities from 1990 to 2005 by country
select l.country,   sum(f.fatalities) as total_fatalities
from fact f, location l, date d
where
f.location_key=l.location_key and f.start_date_key=d.date_key and d.year between
1990 and 2005
GROUP BY l.country
ORDER BY l.country ASC;
```

	country text	total_fatalities bigint
1	Canada	838
2	France	0
3	Saint Pierre and Mi...	21
4	Saudi Arabia	261
5	United States	78

Slice:

--average population affected (in thousands) by types in disaster in the the fall season
select d.disaster_type, CAST(AVG(f.population/1000) as int) as average_population_affected_in_thousands, CAST(AVG(f.evacuated) as int) as average_evacuted
from fact f, disaster d, date dt
WHERE dt.season_canada='fall' and f.disaster_key = d.disaster_key and f.start_date_key=dt.date_key
group by (d.disaster_type)
order by average_population_affected_in_thousands DESC

--average population affected (in thousands) by types in disaster in the the winter season
select d.disaster_type, CAST(AVG(f.population/1000) as int) as average_population_affected_in_thousands, CAST(AVG(f.evacuated) as int) as average_evacuted
from fact f, disaster d, date dt
WHERE dt.season_canada='winter' and f.disaster_key = d.disaster_key and f.start_date_key=dt.date_key
group by (d.disaster_type)
order by average_population_affected_in_thousands DESC

--average population affected (in thousands) by types in disaster in the the summer season

```sql
select  d.disaster_type, CAST(AVG(f.population/1000) as int) as
average_population_affected_in_thousands, CAST(AVG(f.evacuated) as int) as
average_evacuted
from fact f, disaster d, date dt
WHERE dt.season_canada='summer' and f.disaster_key = d.disaster_key and
f.start_date_key=dt.date_key
group by (d.disaster_type )
order by average_population_affected_in_thousands DESC

--average population affected (in thousands) by types in disaster in the the spring
season
select  d.disaster_type, CAST(AVG(f.population/1000) as int) as
average_population_affected_in_thousands, CAST(AVG(f.evacuated) as int) as
average_evacuted
from fact f, disaster d, date dt
WHERE dt.season_canada='spring' and f.disaster_key = d.disaster_key and
f.start_date_key=dt.date_key
group by (d.disaster_type )
order by average_population_affected_in_thousands DESC
```

Example result for winter season:

	disaster_type text	average_population_affected_in_thousands integer	average_evacuted integer
1	Bomb Attacks	4262	0
2	Winter Storm	3956	45
3	Epidemic	2181	[null]
4	Storm - Unspeci...	2134	0
5	Flood	1644	276
6	Marine	1194	2400
7	Hurricane / Typ...	972	934
8	Storms and Sev...	948	41
9	Kidnapping / Mu...	836	0
10	Residential	823	217
11	Shootings	812	0
12	Non-Residential	534	0
13	Derailment Rele...	185	56478
14	Tornado	176	0
15	Leak / Spill Rele...	98	850
16	Manufacturing / ...	80	0
17	Vehicle Release	80	0
18	Landslide	54	224
19	Rail	44	60
20	Air	36	0
21	[illegible]	[illegible]	[illegible]

Dice:

```
-- contrast the the total number of fatalities in Ontario and Quebec from 2010 to 2016
SELECT
  l.province,
  SUM(f.fatalities) as total_fatalities
FROM fact f, date d, location l
WHERE (l.province = 'Ontario' OR l.province = 'Quebec')
AND d.year between 2010 and 2016
and f.start_date_key = d.date_key and f.location_key=l.location_key
GROUP BY province
```

	province text	total_fatalities bigint
1	Ontario	7
2	Quebec	18

Top N:

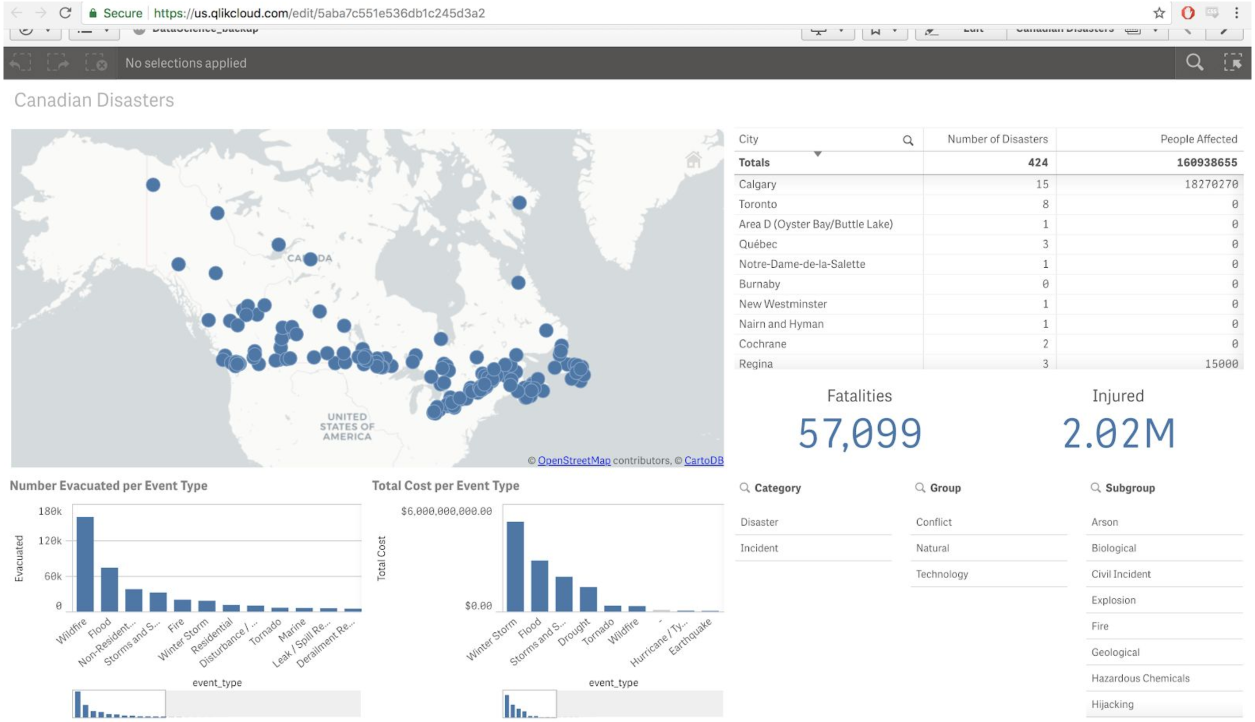
```
-- determine the 5 cities in Canada with the most floods.
SELECT l.city, count(*) as riot_count
FROM fact
INNER JOIN location l ON fact.location_key = l.location_key
INNER JOIN disaster d ON fact.disaster_key = d.disaster_key
WHERE disaster_type = 'Flood' AND country = 'Canada'
GROUP BY city
ORDER BY count(*) DESC
LIMIT 5;
```

Result:

	city text	riot_count bigint
1	Toronto	25
2	Lower Mainland	19
3	Winnipeg	19
4	Red Deer	13
5	Saskatoon	12

Business Intelligence Dashboard:

For the BI dashboard we opted to use Qlik Sense. We decided to showcase the highlight of our data through the various charts. Since we were able to retrieve the coordinates where each disasters occured it allowed us to populate the interactive map as well.



Classification Algorithm:

We employed the decision tree classification algorithm to explore the data. Our goal was to build two models in order to gather knowledge and see if we could determine the disaster type and the disaster subgroup with our data. We employed WEKA with the J48 classifier.

Both of these models were built using the following attributes:

```
Attributes:   29
              fatalities
              injured
              evacuated
              province
              description
              population
              disaster_type
              disaster_subgroup
              disaster_group
              disaster_category
              magnitude
              utility_people_affected
              day
              month
              year
              weekend
              season_canada
              estimated_total_cost
              normalized_total_cost
              federal_dfaa_payments
              provincial_dfaa_payments
              provincial_payments
              municipal_cost
              insurance_payments
              ogd_costs
              ngo_payments
              keyword1
              keyword2
              keyword3
Test mode:    10-fold cross-validation
```

Disaster subgroup:

Using the 10 folds cross-validation test option we obtained great results as were able to successfully classify the instances 98.8272% of the time.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        1264               98.8272 %
Incorrectly Classified Instances        15                1.1728 %
Kappa statistic                          0.9725
Mean absolute error                      0.0023
Root mean squared error                  0.0392
Relative absolute error                  3.4437 %
Root relative squared error             21.6196 %
Total Number of Instances             1279
```

Looking at the confusion matrix we can see how our data follows very closely the diagonal hence the extremely high percentage. This validates the data we chose for this model was good.

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   k   l   m   <-- classified as
  29  10   0   0   0   1   0   0   0   0   0   0   0 |   a = Fire
   9  19   0   0   0   0   0   0   0   0   0   0   0 |   b = Explosion
   0   0  15   0   0   0   0   2   0   0   0   0   0 |   c = Infrastructure failure
   0   0   0  45   1   0   0   0   0   0   0   0   0 |   d = Geological
   0   0   0   0 896   0   0   0   0   0   0   0   0 |   e = Meteorological  Hydrological
   0   0   0   0   0  48   0   0   0   0   0   0   0 |   f = Transportation accident
   0   0   0   0   2   0  17   0   0   0   0   0   0 |   g = Biological
   0   0   0   0   0   0   0  69   0   0   0   0   0 |   h = Hazardous Chemicals
   0   0   0   0   0   0   0   0   7   0   0   0   0 |   i = Arson
   0   0   0   0   0   0   0   0   0   4   1   0   0 |   j = Civil Incident
   0   0   0   0   0   0   0   0   0   0  12   0   0 |   k = Terrorist
   0   0   0   0   0   0   0   0   0   0   0   3   0 |   l = Hijacking
   0   0   0   0   0   0   0   1   0   0   0   0   0 |   m = Space Event
```

Disaster type:

Using the same tests options and data we created a model to determine the disaster type.
In this case the results were not as high as the previous model but were still informative. We were able to correctly classify the instances 807/1279 which is 64% of the time.

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         807               63.0962 %
Incorrectly Classified Instances       472               36.9038 %
Kappa statistic                          0.5624
Mean absolute error                      0.0211
Root mean squared error                  0.1145
Relative absolute error                 50.379  %
Root relative squared error             79.331  %
Total Number of Instances             1279
```

The confusion matrix was quite large but here's most of it:

=== Confusion Matrix ===

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	aa	ab
31	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	102	6	0	0	0	54	1	0	1	5	3	1	0	0	0	0	0	2	0	0	0	2	0	0	0
0	0	0	7	32	0	0	0	33	0	0	1	1	1	0	0	0	0	0	0	2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0
0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	25	10	0	0	0	282	2	0	2	2	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	3	0	0	0	20	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0
0	0	0	2	1	0	0	0	9	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	7	1	0	0	0	55	0	0	0	36	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0
0	0	0	5	1	0	0	0	22	0	0	0	1	22	0	0	0	0	0	0	1	0	0	0	1	0	0	0
0	0	0	3	1	0	0	0	2	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	16	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	16	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	20	0	0	0	0	0	0	0	0
0	0	0	3	1	0	0	0	13	1	0	0	1	3	0	0	0	0	0	0	5	0	0	0	1	0	0	0
0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	3	0	0	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0
0	0	0	3	1	0	0	0	12	1	0	0	1	2	0	0	0	0	0	0	0	0	0	0	21	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5

Just like the previous model we can see how our data follows fairly closely the diagonal hence the high percentage. We can accept the usefulness of this model as the results are satisfactory.

Cluster Analysis:

We employed the expectation-maximization clustering algorithm to explore the data and find some clusters within our data. To do so, we decided to employ WEKA.

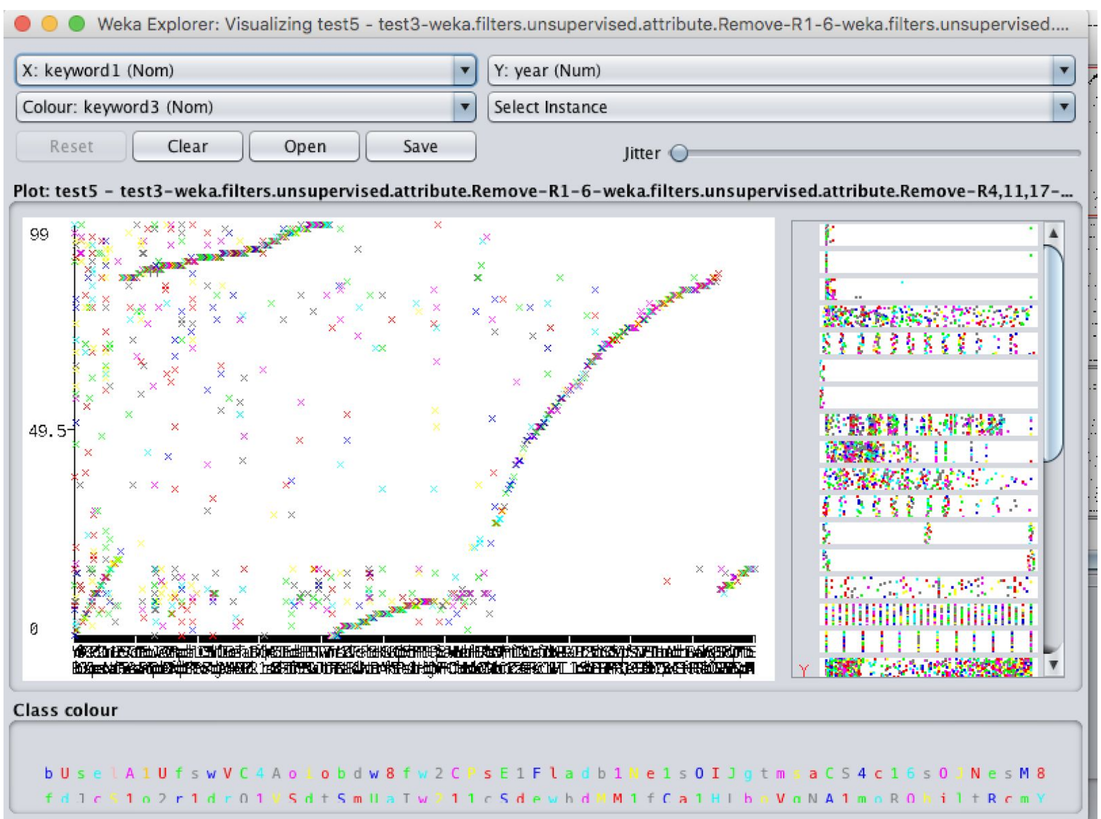
The most obvious cluster we found was within seasons. We can perfectly see how each of the seasons instances are grouped together. Naturally this is a result we expected but it's nice for a first validation of the EM algorithm.

```
                Cluster
Attribute             0      1      2      3
                 (0.14) (0.27) (0.27) (0.32)
==================================
season_canada
  spring              1      1    323      1
  winter              1    321      1      1
  summer              1      1      1    385
  fall              166      1      1      1
  [total]           169    324    326    388
```

Another cluster we found exploring the data was between year and keyword 1.
The statistics and visualization are given below. As we can see we can clearly see the three clusters the algorithm mentioned on the visualization.

```
                           Cluster
Attribute                        0         1         2
                            (0.24)    (0.42)    (0.33)
====================================================
year
  mean                     58.7398    9.0144   88.7654
  std. dev.                16.8604    4.9382    6.9083

keyword1
  fire                       3.168   10.9864    5.8456
  BC,                       14.209   10.6027    6.1882
  Wellington                1.0059    2.9941         1
  buried                     1.001     1.999         1
  September                 2.0599    1.9943    1.9458
  Calgary                   1.0505    1.9978    5.9517
  Arrow                      1.001     1.999         1
  19,                       1.0009    1.9991         1
  February                  5.8651    2.9928    2.1422
  swept                     2.0018    1.9981         1
  Columbia,                 2.9374    2.2257    3.8369
  Quebec                    2.4535    8.9548    1.5918
  side.                     1.0012    1.9988         1
  caused                    3.3756   13.9668   11.6576
  Provinces,                3.8772    5.1076    1.0153
  passenger                  2.001     1.999         1
```



Anomaly Detection:

In order to detect anomalies we opted to use statistical tests.

Appendix A

```
CREATE TYPE season as ENUM('winter','spring','summer','fall');

CREATE TABLE project.Date (
  date_key int PRIMARY KEY,
  day int,
  month int,
  year int,
  weekend boolean,
  season_canada season
);

CREATE TABLE project.Disaster (
  disaster_key int PRIMARY KEY,
  disaster_type text,
  disaster_subgroup text,
  disaster_group text,
  disaster_category text,
  magnitude numeric,
  utility_people_affected int
);

CREATE TABLE project.Summary (
  description_key int PRIMARY KEY,
  summary text,
  keyword1 text,
  keyword2 text,
  keyword3 text
);

CREATE TABLE project.Location (
  location_key int PRIMARY KEY,
  city text,
  province text,
  country text,
  canada boolean,
  longitude numeric,
  latitude numeric,
  description text
);

CREATE TABLE project.Costs (
  costs_key int PRIMARY KEY,
  estimated_total_cost numeric,
  normalized_total_cost numeric,
  federal_dfaa_payments numeric,
  provincial_dfaa_payments numeric,
```

```
  provincial_payments numeric,
  municipal_cost numeric,
  insurance_payments numeric,
  ogd_costs numeric,
  ngo_payments numeric
);

CREATE TABLE project.Fact (
  start_date_key int,
  end_date_key int,
  location_key int,
  disaster_key int,
  description_key int,
  costs_key int,
  fatalities int,
  injured int,
  evacuated int,
  population int
)
```